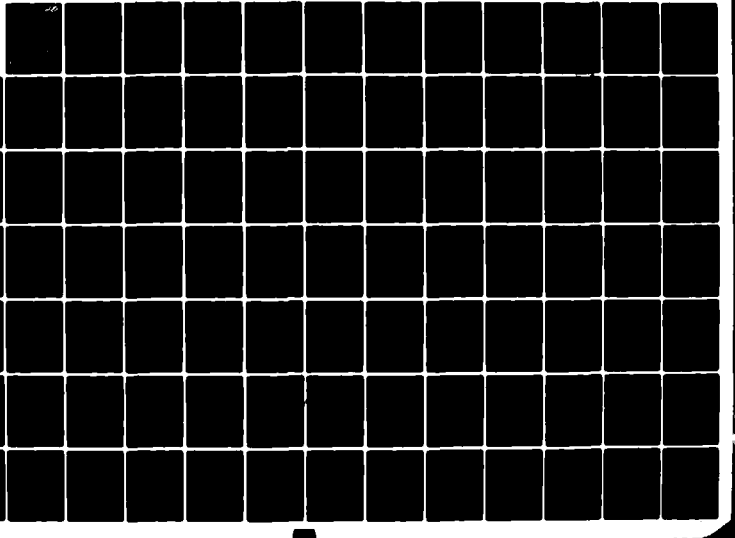
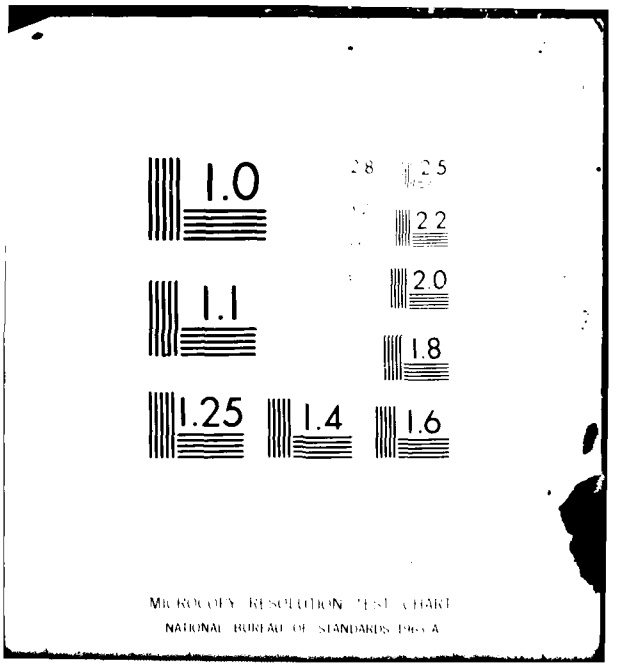
AD-A110 069 GEORGIA INST OF TECH ATLANTA SCHOOL OF ELECTRICAL EN--ETC F/G 9/2
STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USIN--ETC(U)
JUL 81 J L HAMMOND, J H SCHLAG, D K LAM DAAG29-80-K-0009
UNCLASSIFIED E21-616 NL

1 OF 2
AD A
110069




1.0
1.1
1.25
1.4
1.6
2.0
1.8
2.2
2.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

TECHNICAL REPORT

TASK REPORT E21-616

STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USING MEASURED DATA AND ANALYTICAL/SIMULATION MODELS

J. L. HAMMOND

J. H. SCHLAG

Submitted To

ARMY RESEARCH OFFICE

JULY 1981

School of Electrical Engineering

GEORGIA INSTITUTE OF TECHNOLOGY

Atlanta, Georgia 30332

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: E21-616	2. GOVT ACCESSION NO.: AD-A110 069
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Study of a Heterogeneous Distributed Microcomputer Network Using Measured Data and Analytical/ Simulation Models	5. TYPE OF REPORT & PERIOD COVERED: Annual
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): J. L. Hammond, J. H. Schlag, D. K. Lam, D. N. Murray, P. J. P. O'Reilly	8. CONTRACT OR GRANT NUMBER(s): DAAG29-80-K-0009
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Georgia Institute of Technology, School of Electrical Engineering, Atlanta, Georgia 30337	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: US Army Research Office, P. O. Box 12211, Research Triangle Park, NC 27700	12. REPORT DATE: July, 1981
	13. NUMBER OF PAGES: 155
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): US Army Computer Systems Command, Attn: CSCS-ATA, AIRMICS, 115 O'Keefe Bldg., Georgia Institute of Technology, Atlanta, Georgia	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Local Computer Networks
Network test Laboratory
Network Performance Monitoring
Network Simulations
Network Models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

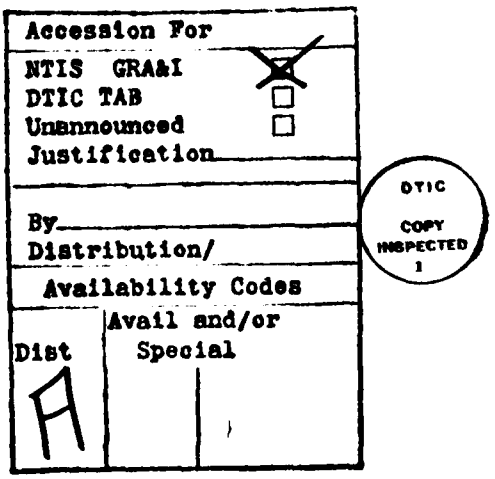
Block 20

This report is the documentation for the contract, "Study of a Heterogeneous Distributed Microcomputer Network Using Measured Data and Analytical/Simulation Models," done as a joint project between Georgia Institute of Technology (Georgia Tech) and the Army Institute for Research in Management and Information Sciences (AIRMICS) over the one-year period of June 15, 1980 to June 15, 1981.

This study has been concerned with exploring the characteristics of a network with potential application in management information systems using both analytic/simulation models (developed in the study) and measured data obtained from the network models. The network used has been constructed in the School of Electrical Engineering at Georgia Tech with support from AIRMICS. A significant feature of the proposed program has been the approach of dealing with both experimental data from the physical network and results from analytical/simulation models.

The completed study has been directed toward an enhanced understanding of heterogeneous, distributed microprocessor networks. Monitor equipment has been studied, installed and used to obtain experimental data on the operation of the network and analytical/simulation models were tailored to describe as accurately as possible the operation of the actual network.

The results of the study are models which accurately describe network behavior. These models, which depend explicitly on well-defined parameters, provide the facility to investigate alternative designs and to predict network behavior for many types of inputs, such as would be specified for particular management information systems.



ACKNOWLEDGMENTS

The work reported herein was performed under a grant from the U. S. Army Research Office in support of work for the U. S. Army Computer System Command Institute for Research in Management Information and Computer Sciences. The study was one task of a project entitled "Study of a Hetergeneous Distributed Microcomputer Network Using Measured Data and Analytical/Simulation Models."

Principal Investigators: J. L. Hammond

J. H. Schlag

Contributors: D. K. Lam

D. N. Murray

P. J. P. O'Reilly

L. S. Osoinach

H. Y. So

K. S. Zimler

TABLE OF CONTENTS

TABLE OF CONTENTS

(Continued)

1. INTRODUCTION

A. Introduction

This report is the documentation for the contract, "Study of a Heterogenous Distributed Microcomputer Network Using Measured Data and Analytical/Simulation Models," (Contract No. DAAK70-79-D-0087) done as a joint project between Georgia Institute of Technology (Georgia Tech) and the Army Institute for Research in Management and Information Sciences (AIRMICS) over the one-year period of June 15, 1980 to June 15, 1981.

In the past decade or so, a number of computer-communication networks have come into being that allow the joint sharing of the resources of a number of computers. The advent of microcomputers has accelerated the potential gains to be achieved from distributed computer networks, but at the same time has necessitated some changes in the ground rules for designing computer networks and has introduced new design problems arising from the attempt to add significantly improved features.

This study has been concerned with exploring the characteristics of a network with potential application in management information systems using both analytic/simulation models (developed in the study) and measured data obtained from the network models. The network used has been constructed in the School of Electrical Engineering at Georgia Tech with support from

AIRMICS and is described in detail later in this report. A significant feature of the proposed program has been the approach of dealing with both experimental data from the physical network and results obtained from analytical/simulation models.

B. Objective

The completed study has been directed toward an enhanced understanding of heterogenous, distributed microprocessor networks. Monitor equipment has been studied, installed and used to obtain experimental data on the operation of the network and analytical/simulation models have been developed. The analytical/simulation models were tailored to describe as accurately as possible the operation of the actual network.

The results of the study are models which accurately describe network behavior. These models, which depend explicitly on well-defined parameters, provide the facility to investigate alternative designs and to predict network behavior for many types of inputs, such as would be specified for particular management information systems.

The study required four tasks:

Task 1: Design and Implementation of Non-Intrusive Monitoring Equipment

The first task of this study synthesized a limited, cost-effective collection of hardware and software to obtain the

data required to validate analytic/simulation models and characterize network performance. The heart of the hardware monitor is the "monitoring memory", a two port memory which is on both the monitor's bus and the node microprocessor's bus, and which is implemented non-intrusively. By correctly programming the monitor processor, any function in the node may be monitored.

Task 2: Design and Implementation of Traffic Generating Equipment

The second task measured the behavior of the network under various controlled stimuli. To accomplish this, equipment was developed which could generate network traffic messages. This involved both hardware and software development. The hardware was designed to allow the traffic generating machine to communicate with both the network processors and the monitoring processors. Real time software was necessary to drive the hardware and actually implement an experiment.

Task 3: Development of Analytical/Simulation Models

The third task was the development of the analytical/simulation model for the AIRMICS/Georgia Tech experimental network. The model represents the computer-communication system and, to some extent, the host microcomputers as a network of queues. Of particular interest were the following performance measurements: 1) end-to-end delay for the individual elements of the communication network, 2) channel utilizations, 3) throughput for all elements of the

network and hosts, and 4) throughput in terms of job completions for the entire system. Analytical analysis was used to predict the network delays as a function of system loading.

Task 4: Evaluation and Test of the Model with the Experimental Network

The fourth task was the prediction of results of various amounts of network loading based on the theoretical models. The variables accounted for by the queueing network model were measured in the experimental network under controlled conditions. The results of such measurements were then compared to computed results from the model and the model refined to minimize differences. The results of no load and minimal load conditions were verified experimentally; however, hardware limitations prevented the testing of average and full load conditions.

2. Mathematical Models

For the AIRMICS/Georgia Tech Network

2.1 Introduction

This section of the report presents and discusses queueing models for elements of the AIRMICS/Georgia Tech network. It also reviews several approaches that can be used in analytical studies of network configurations.

As discussed in detail elsewhere in this report, the network consists of a maximum of five switching nodes which can be connected in a variety of ways to form a packet switched network. In operation, the network interconnects host computers, terminals and other devices which may be distributed over a relatively large geographical area.

The interest in much of the present study is centered on the performance of the network as measured by such quantities as throughput and delay, which are distinctive properties of the network. Although these quantities depend on the nature of the inputs and outputs of the network, they can be considered to a large extent independent of specific details of the interconnected hosts, etc. in analyzing or designing a network.

Queueing theory models have been used almost exclusively in obtaining mathematical relations for throughput and delay. There is a considerable amount of literature on queueing theory as a mathematical discipline, (see for example [1], [2], and [3]), and in the last decade there have been numerous and significant studies of computer networks, as modeled by interconnected systems of queues, [4], [5], [6]. The literature describing the matching of mathematical models to specific physical systems, however, is relatively sparse [7].

The material concerning mathematical models and performance analysis in this Section is organized as follows: Section 2.2 gives basic queueing models for network components. Several parameters of the queueing models, namely, buffer capacity, the effect of acknowledgements, input and output statistics, and errors are discussed in Section 2.3.

Section 2.4 discusses methods for analysis with the queueing models. A number of techniques based on different types of approximations are discussed in this Section. Section 2.5 gives a concise review of the network models and solution techniques and Section 2.6 lists references. An Appendix concludes the section.

2.2 Queueing Models for the Network Components

The basic elements of the network are the Switching Nodes and the Connnecting Lines or Channels. Connected to the network will be Host Computers, Terminals, and possibly other devices such as Disk Memories, etc.

The interest in the present study is in modeling the network and those aspects of the devices connected to the network which will affect the planned experiments. This will involve modeling Host Computers, used in an elementary data processing role, as well as the basic network elements.

The network switching nodes are microcomputers with Buffer Memories and four I/O lines controlled by the Nodal CPU. Arriving messages queue for the use of the Node CPU. The Node CPU controls instructions for error detection, storing incoming messages in buffers, sending acknowledgement messages to the node originating the message, and choosing the output node and route to destination. The time to process these instructions is called "nodal processing time" and is typically small enough to be neglected in a queueing model.

Messages routed to an output node by the CPU must queue in an output Buffer for the use of the output channel. The "service time" of the output channel is determined by its line capacity using the relation "service time" equals message length divided by line capacity. This service time is included in queueing models to account for the channel. Typically the "Propagation Time," or time for one bit to travel from the sending to the receiving terminal, is small enough to be negligible.

Figure 2.1 gives a basic model of the Switching Node and Output Channels, neglecting nodal processing time and propagation time. Models such as that in Figure 2.1 are discussed, for example, by Kleinrock [8], Wong [4], and Samari [6].

For use in the Experiments discussed below, the Host Minicomputers can be modeled by the single queue shown in Figure 2.2a. A slightly more general model with an additional queue and processor FM (file manager) is hown in Figure 2.2b. The additional processor determines whether a received message requires local processing or is intended for another Host. More general models for Minicomputer Hosts are discussed by many authors, see for example [9] and [10].

2.3 Parameters of the Queueing Models

2.3.1 Buffer Capacity: The capacity of the Buffer Pool at each Switching Node is finite, being 3200 bytes total. In the physical system, the finite buffer capacity leaves open the possibility that some messages will be lost because there will be no buffer in which to store them. The possibility of occasional lost messages could be indicated in the model of Figure 2.1 by adding a dotted "lost traffic" branch to every incoming channel.

To make a first order model tractable for analytic solution, it is convenient to assume that buffers are infinite. This assumption is not absolutely essential and finite buffer models are discussed

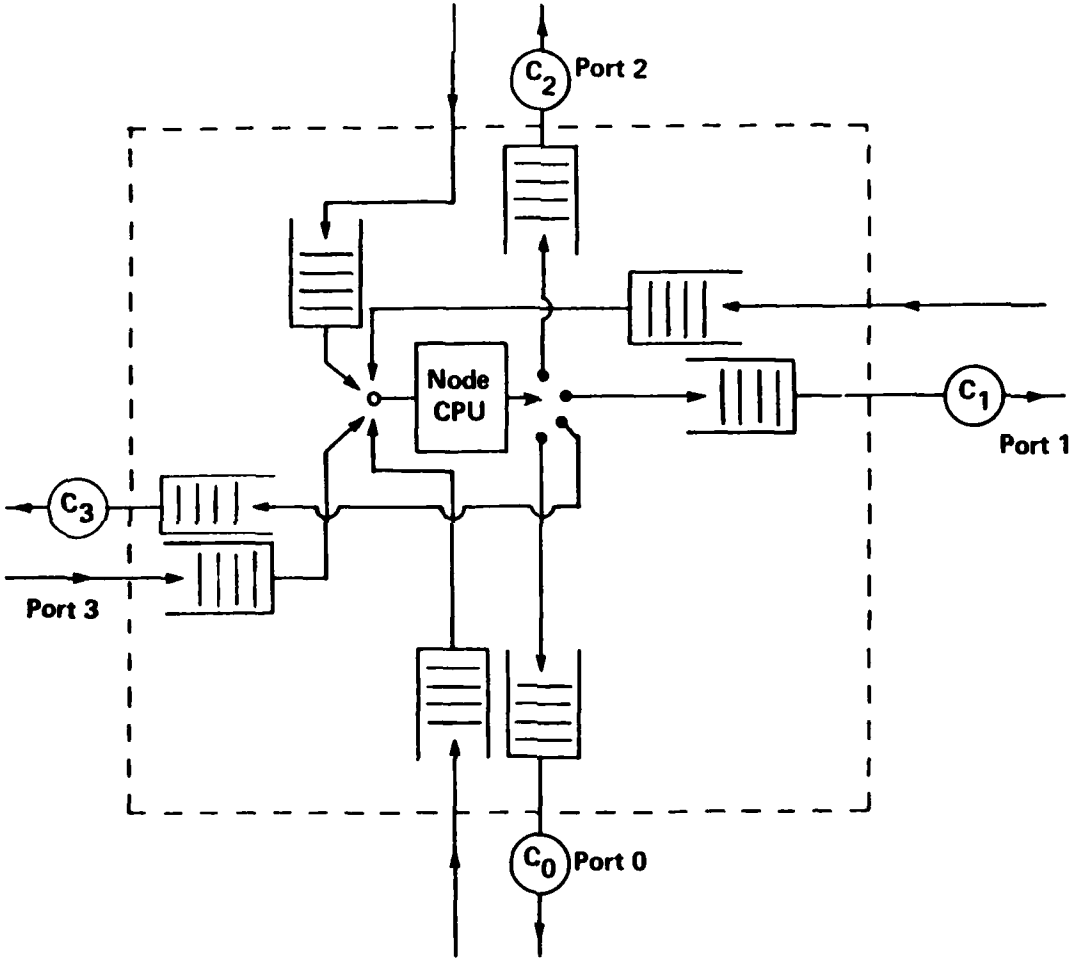
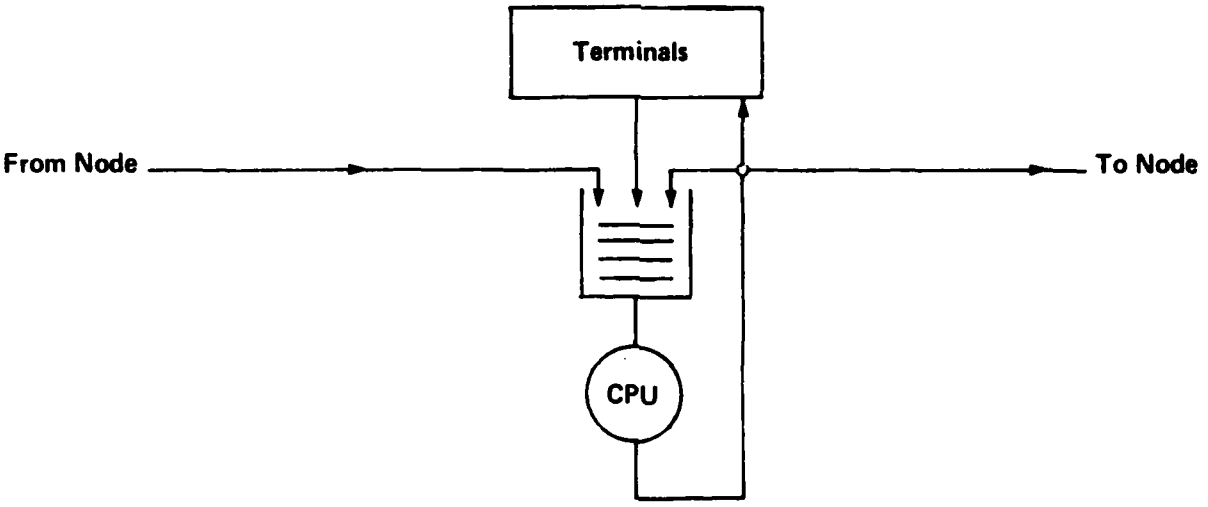
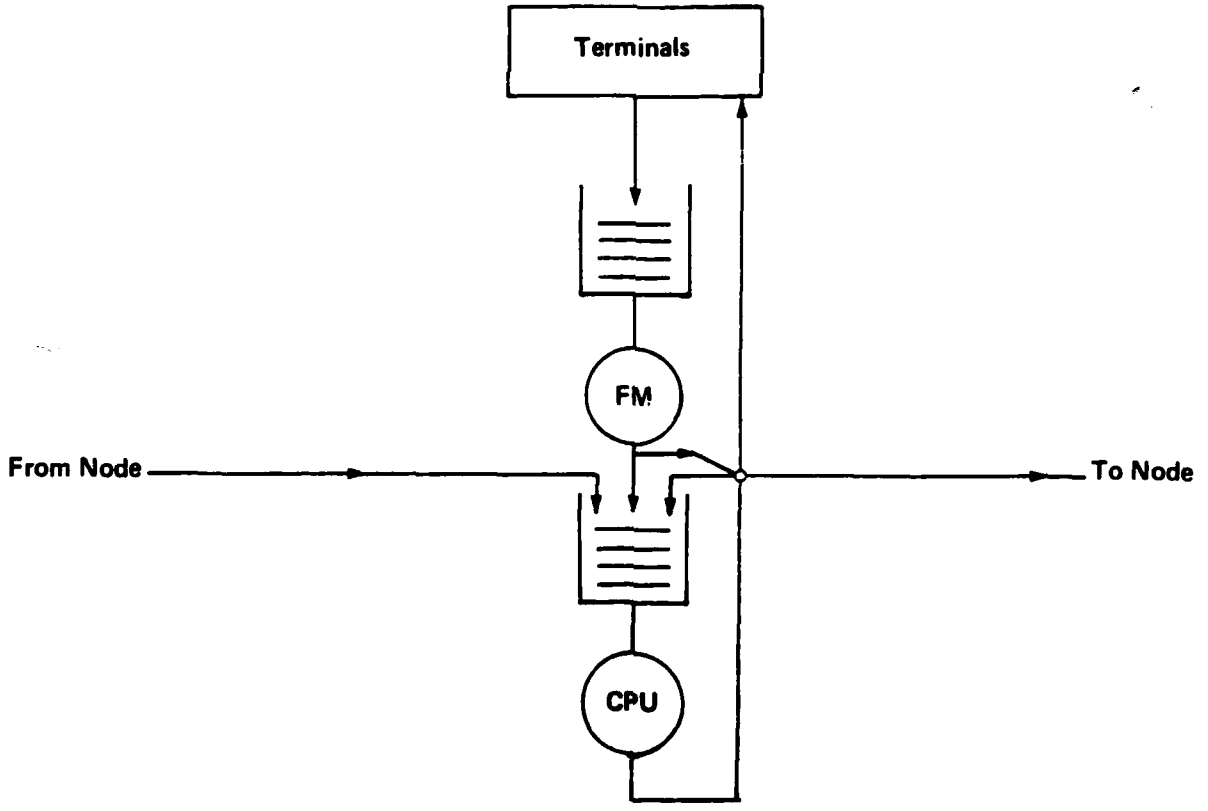


FIGURE 2.1. BASIC MODEL OF SWITCHING NODE AND OUTPUT CHANNELS.



(a)



(b)

FIGURE 2.2. SIMPLIFIED MODELS OF HOST MINICOMPUTER.

below.

2.3.2 Acknowledgements: The network is structured so that positive acknowledgements are exchanged between Nodes. If a message is lost or received in error it is not acknowledged and, after a specified "time-out", the message is retransmitted.

Acknowledgement messages are ten bytes in length and are transmitted between Nodes on paths directed in the reverse direction from the message traffic. In general, acknowledgements will queue with other traffic traveling in its direction between the Nodes.

Acknowledgements impact the queueing models in two ways. The nodal CPU must direct the storing (actually the retention in storage) of replicas of each transmitted message until an acknowledgement is received at which time the retained message is "written over". This requires some part of the nodal processing time, but since the total nodal processing time is usually negligible with respect to other effects, this increment to nodal processing time is typically ignored in the models.

The second impact on queueing is more important. Acknowledgement traffic must be added to other traffic and accounted for in determining queue sizes.

2.3.3 Input/Output Traffic Statistics: In an analysis with queueing models, the statistical distributions of arriving message traffic must be known or assumed. The same comment applies to the time required for message transmission over the connecting channels (i.e., service times).

With respect to arriving traffic, the statistical distribution will, obviously, depend on particular applications. There is, however, very little literature describing measured distributions of message traffic. The data that exists is not in conflict with assuming the most tractable process, namely Poisson, for arriving traffic.

Now consider service times. The number of bits per second processed over a communication channel is constant at the rated capacity of the channel, during the time messages are queued and ready. The length of messages, however, can vary so that the time to process complete messages can be variable. As with arriving message distributions, the statistical distributions of message lengths will vary with the application and little data on typical distributions is available.

Logical choices for the distribution function of message lengths include the geometric distribution, and in fact, the (deterministic) constant message length. An exponential distribution for message lengths is often assumed to give a tractable analytical model.

2.3.4 Random Errors: The network deals with errors through the use of an error detecting code. After a packet has been transferred between two nodes, the overhead bits for error detecting are evaluated and a positive acknowledgement is sent back to the originating node if the packet has been received correctly. If the received packet is not correct, no acknowledgement is sent and, after a time out, the packet is retransmitted.

An exact model of the network would have to account for the finite probability of retransmission occuring due to random bit error. For a well designed network operating over short lines, however, the probability of bit error is low and therefore, retransmissions are not an important factor in network operation. They will not be included in the models to be discussed below.

2.4 Network Studies with Queueing Models

To model a specific network configured from the basic components of the AIRMICS/Georgia Tech network, the basic queueing models must be combined into a network of interconnected queues. The interest in the model centers on the ability to obtain throughputs and delays for the composite network and to relate these quantities to basic network parameters such as input traffic, line capacities, packet lengths, etc.

Two basic approaches are available for studying the models; analytic methods and simulation methods. Both approaches have advantages and disadvantages.

Analytic methods have the very desirable property of yielding explicit relations between the variables and making direct design procedures and design trade-offs possible. Unfortunately, to make analytical methods feasible, a number of approximations must typically be made, and accuracy of approximation becomes a serious question.

Simulation methods, on the other hand, especially direct "Monte Carlo" simulations do not require the degree of approximation of models used for analytic work. The shortcoming of simulation methods, however, lies in the fact that they do not yield explicit relations between the variables. Instead, every set of numerical values for the parameters must be used in separate, complete, calculations. To obtain desired statistics, it is usually necessary to obtain many "runs" for a single set of parameters and employ statistical estimation to obtain the desired results.

Given the properties of the two basic approaches to model analysis, it seems reasonable to use both in any extensive studies. Analytical results, and the explicit relations which they provide, would seem to give essential background or "first cut" information and thus they are obtained first. The present study is limited to the analytical methods.

2.4.1 Models Which Assume Independence Of Queues:

When a message path through a network must pass through one or more Switching Nodes, the queueing model becomes a tandem connection of queues with the output of one feeding into the input of another. Analytic methods for tandem queues are severely restricted and tractable results require the equivalent of the following assumptions [11], [12]:

1. The external network arrivals follow a Poisson process
2. Each time a message enters a new node it is assigned a new length selected from an exponential distribution; all message classes have the same message length distribution
3. The routing algorithm is fixed.
4. All buffers have infinite capacity.
5. The queuing discipline is first come - first served.

Kleinrock [8], for example, discusses these assumptions in detail and shows that when they apply, each node can be analyzed independently as an M/M/1 queue*. The total average network delay is then a simple weighted sum of the independently calculated nodal delays.

In spite of the severity of these assumptions, several authors [13], [8] have noted that in treating many effects and for most networks, the answers are sufficiently accurate for engineering purposes.

---

*M/M/1 is a common notation for Poisson arrivals, exponential service, and one server.

The basic M/M/1 equation for the average time spent waiting for and using the ith channel is given by:

$$T_i = \frac{1}{\mu C_i - \lambda_i} \quad , \tag{2.1}$$

where λ_i is the traffic rate in the ith channel, C_i is its capacity, and $1/\mu$ is the average message length.

This equation can be adapted to the case of combined data traffic and control traffic. The average waiting time is due to both types of traffic, while the service time is due only to the data traffic. This yields:

$$T_i = \frac{\lambda_i/\mu' C_i}{\mu' C_i - \lambda_i} + \frac{1}{\mu C_i} \tag{2.2}$$

where $1/\mu'$ is the average message length of all packets and $1/\mu$ is the average length of data packets.

If nodal processing time is denoted as K, and propagation time on the ith channel as P_i, then Kleinrock gives the total average message delay, T, for the whole network as:

$$T = K + \sum_{i=1}^{m} \frac{\lambda_i}{\gamma} \left[\frac{\lambda_i/\mu' C_i}{\mu' C_i - \lambda_i} + \frac{1}{\mu C_i} + P_i + K \right] \tag{2.3}$$

where γ is the total traffic entering the network in messages/second, and m is the number of inter-nodal links.

The simplicity of (2.3) makes it possible to obtain approximate results in a straightforward manner even for complicated network structures.

2.4.2 Models with Finite Buffer Space: Although the tractable models of Section 2.4.1 will solve many network problems, there are several effects which are directly attributed to finite buffer space. These effects are lost messages due to full buffers, congestion, deadlock and other flow problems. It is, of course, desirable to have models, amenable to analytic solution, which can approximate these effects.

Several approaches from the literature will be reviewed after some comments on the general problems.

Loss of packets can occur, for example, if a packet attempts to enter a node in which all available buffer space for the required output channel is full. The packet will be blocked from entering the node and thus will be stored in the previous node until buffer space becomes available, (thereby propagating congestion back through the network), or may be dropped after being locally acknowledged, (thereby requiring retransmission from the originating node). The general effect of packet loss and other congestion in the network is to cause a drop in throughput as illustrated in Figure 2.4.1.

Deadlock is a more severe flow control problem in which the throughput for a particular link goes to zero. This may occur when

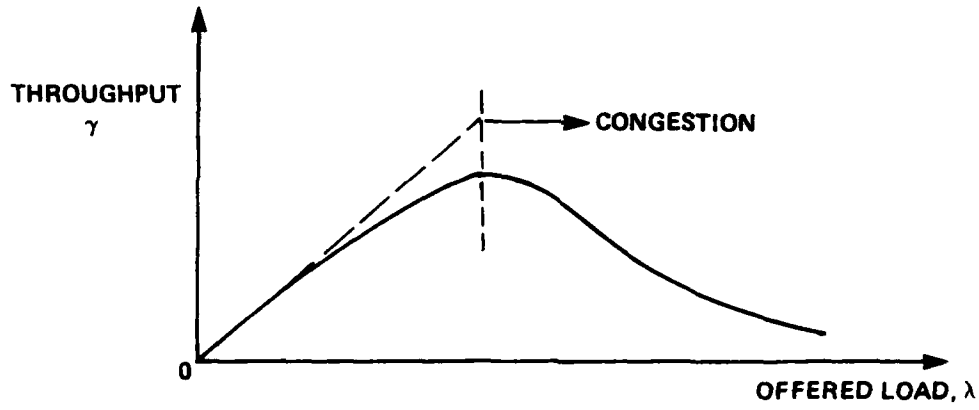


FIGURE 2.4.1. THROUGHPUT CHARACTERISTIC WITHOUT FLOW CONTROL.

two nodes wishing to communicate have all the buffers allocated to the required link full. Reassembly lock-up is another finite storage problem which may occur in the destination switching node where, in many networks, message reassembly takes place.

Finite buffer queueing models for switching nodes have been developed and used with considerable success in buffer management and local flow control problems. The models retain the assumptions listed in Section 2.4.1 for the most tractable models, with the exception of the infinite buffer (assumption 4). Irland [14] and Lam [15] both have analyzed models for a single node with finite buffers. Their results are summarized as follows: Consider a Poisson packet arrival stream with average rate λ (the offered load) arriving at a node, capable of holding at most N packets, see Figure 2.4.2. Let C be the transmission rate of the single output channel and μ^{-1} the average packet length. These incoming packets are blocked (and dropped from the network) with probability P_B, the probability of all the buffers being occupied, so that the throughput, γ, is given by:

$$\gamma = \lambda (1-P_B) \quad . \tag{2.4}$$

The effect of nodal blocking at the next stage of a tandem connection is to reduce the effective channel capacity to $C(1-P_B)$ assuming an identical blocking probability. Under this assumption, the queueing model for a channel becomes effectively a finite M/M/1 queue for which the blocking probability is given by:

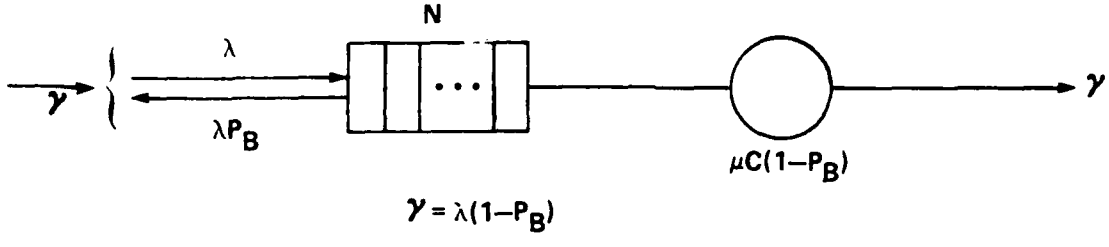


FIGURE 2.4.2. MODEL OF A FINITE STORAGE NODE.

$$P_B = \frac{(1-\rho)\rho^N}{1-\rho^{N+1}} \qquad (2.5)$$

where

$$\rho = \frac{\lambda}{\mu C(1-P_B)} \qquad (2.6)$$

Combining (2.4), (2.5), and (2.6) gives the equation for a throughput-offered load characteristic such as that of Figure 2.4.1.

In a more general model of a node, such as that of Figure 2.4.3 there are a number of output channels to other nodes and perhaps a link to a local host as well as an input buffer. Such a node also deals with locally generated traffic, traffic coming from similar switching nodes (often termed transit traffic), and positive acknowledgements and timeouts. Irland [14] showed that restricted buffer sharing policies improve the throughput under congestion conditions. Lam [15] developed an algorithm for determining the nodal buffer requirements so as to minimize the loss probabilities and thereby balance the traffic among the output channels. Their work was extended by Lam and Reiser [16] who showed that input buffer limits could be used effectively as a form of congestion control.

Penotti and Schwartz [17] considered a number of finite storage nodes in tandem along a virtual circuit such as that shown in Figure 2.4.4. They model the effect of external arrivals by reducing the

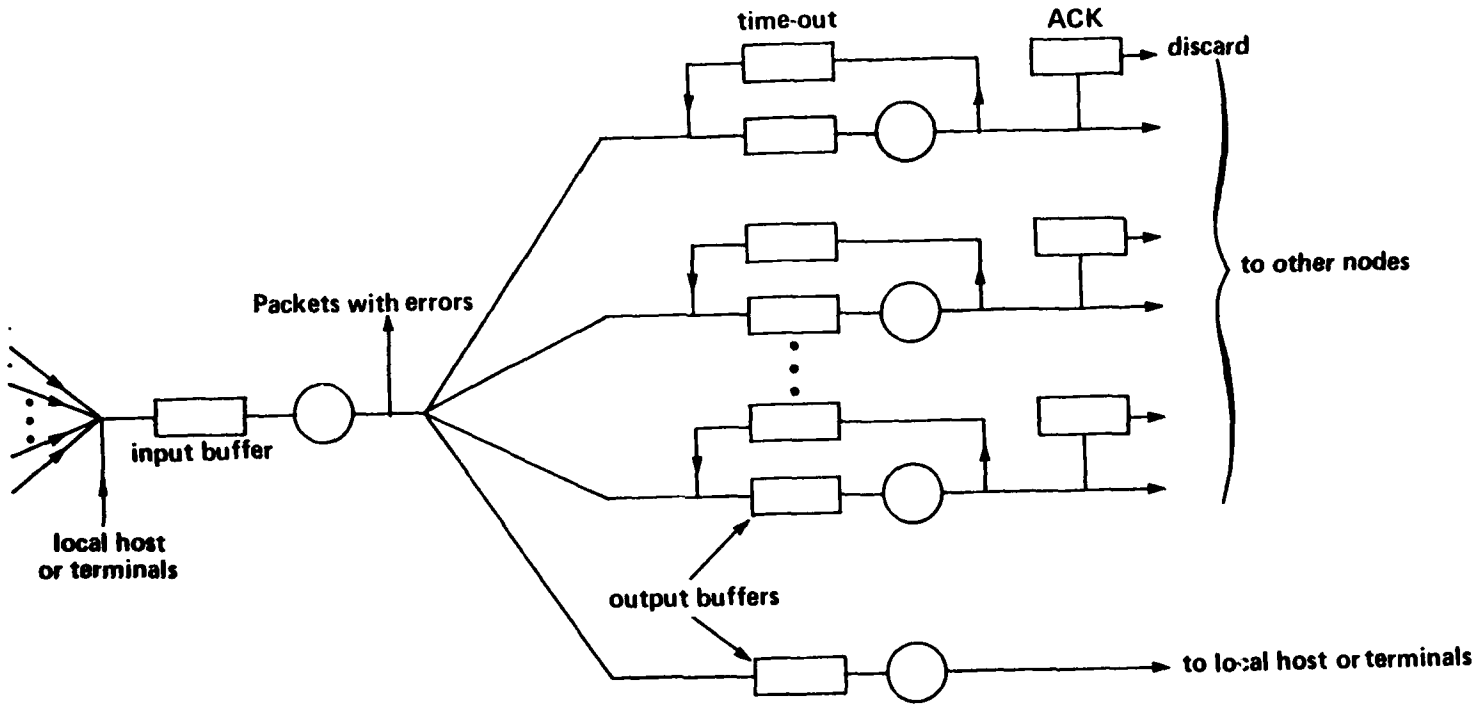


FIGURE 2.4.3. NODE MODEL.

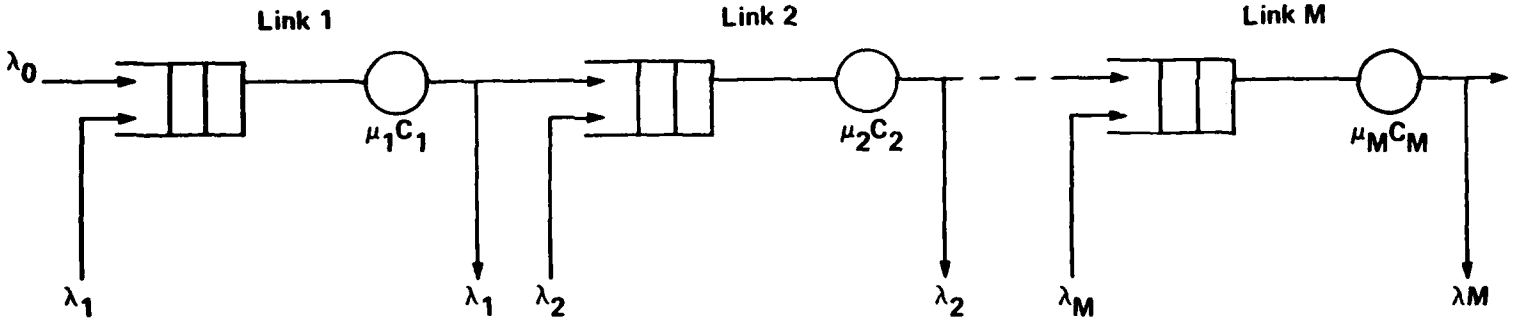


FIGURE 2.4.4. QUEUEING MODEL OF A VIRTUAL CIRCUIT.

link capacity by the external packet arrival rate; i.e. for link i, the effective capacity becomes:

$$\mu_{ei} = \mu_i C_i - \lambda_i \qquad (2.7)$$

where λ_i is the external packet arrival rate to link i. This is equivalent to reducing the network of Figure 2.4.4 to that of Figure 2.4.5. If (2.4) is used to express the effective service rate for the ith link, the result is:

$$\mu_i' = \mu_{ei} \ (1-P_{B_{i+1}}) \quad , \qquad (2.8)$$

Where P_{Bi} is the blocking probability for the ith link.

The overall blocking probability for the complete circuit of Figure 2.4.5 can be evaluated iteratively working backwards from the final node. The overall throughput is then found to be:

$$\gamma = \lambda_0 \ (1-P_{B1}) \qquad (2.9)$$

where λ_0 is the load offered to the virtual circuit by the source.

Simulation results for local control have shown good agreement with this approximate model. Rudin [18] has extended the work of Pennotti and Schwartz to study the phenomenon of reduced throughput with increased offered load.

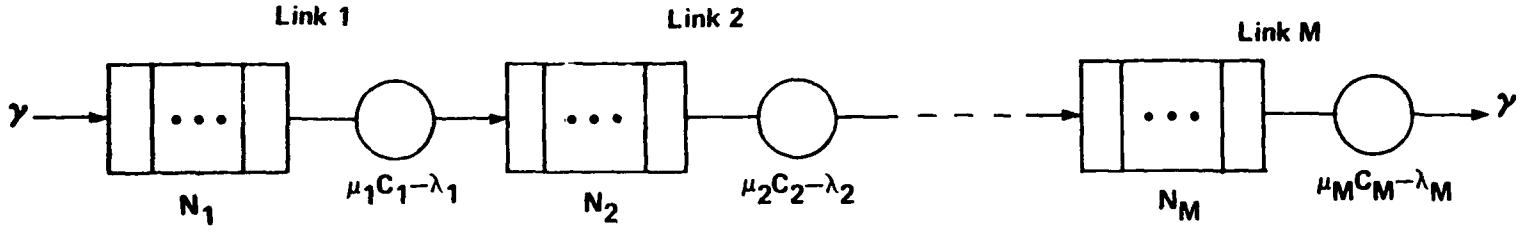


FIGURE 2.4.5. APPROXIMATE VIRTUAL CIRCUIT MODEL.

A more powerful approach to modeling finite buffer storage networks than those discused above involves the use of a Markov chain to model the complete network. This method potentially removes most of the restrictions listed in Section 2.4.1. Some work has been found in the literature using this method. For example, a recent paper [19] discusses a two node model which gives a six or eight dimensional state probability vector. This model incorporates restrictions to the buffer and window sizes as well as differentiating between local and "foreign" traffic; acknowledgements are also included in the model. The author uses sparse matrix algorithms to attack the large dimensional matrices which must be inverted.

Although the paper cited, and a few others provide an approach to the Markov chain method, they do not result in practical methods.

Since there is a need for more accurate methods of dealing with the finite buffer problem, some time was spent on the present project in attempting to develop an alternate Markov chain approach. This work is discussed in the next Section.

2.4.3 A New Approach for Finite Buffer Models Using Markov Chains: If one continuous time Markov chain is used to model the complete network, each state in the chain will have a value equal to the number of packets (queued and being served) on a particular channel of the network. Since the total storage at each

node is finite, the Markov chain is finite valued, although in general, the number of states is extremely large. The general technique to solve such a Markov chain is to write down the local balance equations [1] and to solve them for the state probabilities $P(n_1, n_2, \ldots n_n)$, i.e., the probability of having n_1 packets in queue 1, $\ldots$ n_n packets in queue n. Such a method is tedious and costly, as the inversion of an extremely large matrix is required.

An alternative graphical method of solving for the state probabilities in a Markov chain is proposed using a state Reduction Method developed by Liu [20]. Liu's method is developed for discrete time Markov chains and gives the mean first passage time from one specified state to another. Work on the present project has extended the general method to continuous time Markov chains as discussed in an Appendix. From the mean first passage times, the state probabilities and other parameters of interest, such as average delay time, throughput and average number of packets stored at each node, have been evaluated.

To illustrate the method devised, consider a queueing model with the following characteristics:

1. The network is "closed" so that a fixed number of messages are circulating.
2. Service times are exponentially distributed.
3. There is a maximum to the number of messages in the network such as, for example, might be imposed by window flow control.

Assumption 3 insures that the storage used is finite and hence both finite and infinite buffers give the same results. A specific example from the above class is used to illustrate the method.

Consider a closed queueing network with 3 queues in tandem and with 2 messages circulating such as shown in Figure 2.4.6. All queues can be assumed to have infinite capacity.

The messages can be distributed throughout the network in six possible ways, so that the Markov chain has six states. Each of the service times is exponentially distributed with means $1/\mu_1$, $1/\mu_2$, and $1/\mu_3$ respectively. The continuous time Markov chain can be represented by the transition rate diagram of Figure 2.4.7. Kleinrock [1] and other texts on Markov chains show how this diagram is developed.

In the diagram, the three digit numbers in the "state circles" identify the states by giving the number of messages in queue 1, the number of messages in queue 2, and the number of messges in queue 3 in the order stated. These queue occupancies are denoted n_1 , n_2 , and n_3 respectively. For the network defined above,

$$n_1 + n_2 + n_3 = 2. \tag{2.10}$$

The standard approach to determining the steady state probabilities is to solve the matrix equation, [1],

$$\pi Q = 0 \tag{2.11}$$

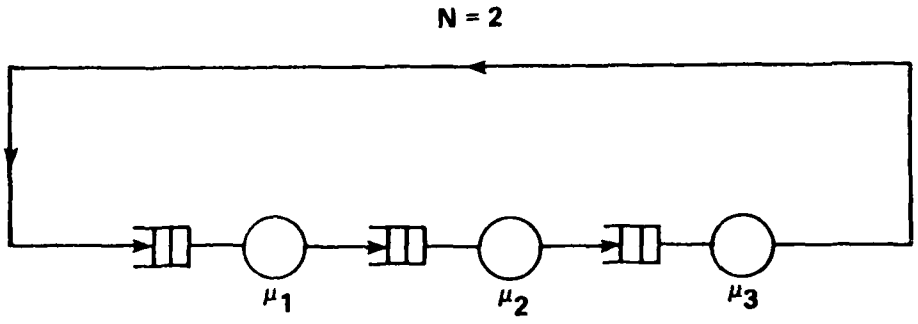


FIGURE 2.4.6. CLOSED QUEUEING NETWORK WITH 3 QUEUES AND 2 CIRCULATING MESSAGES.

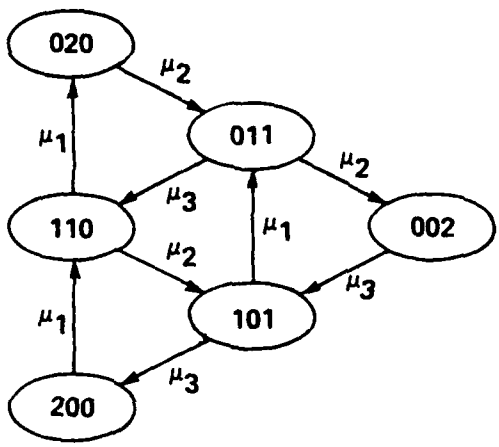


FIGURE 2.4.7. STATE–TRANSITION–RATE DIAGRAM FOR QUEUEING NETWORK OF FIGURE 2.4.6.

subject to the constraint

$$\sum_{i=1}^{6} \pi_i = 1 \quad . \tag{2.12}$$

In (2.11), $\pi = [\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6]$ is a vector of state probabilities, π_i, and Q is the transition-rate matrix

$$Q = \begin{bmatrix} -\mu_2 & 0 & 0 & \mu_2 & 0 & 0 \\ \mu_1 & -(\mu_1+\mu_2) & 0 & 0 & \mu_2 & 0 \\ 0 & \mu_1 & -\mu_1 & 0 & 0 & 0 \\ 0 & \mu_3 & 0 & -(\mu_2+\mu_3) & 0 & \mu_2 \\ 0 & 0 & \mu_3 & \mu_1 & -(\mu_1+\mu_3) & 0 \\ 0 & 0 & 0 & 0 & \mu_3 & -\mu_3 \end{bmatrix} \tag{2.13}$$

For example, if $\mu_1 = 3$, $\mu_2 = 2$, $\mu_3 = 1$, the matrix equation (2.11) yields the solution

$$\pi_1 = p_{020} = 9/85 \qquad \pi_4 = p_{011} = 18/85$$

$$\pi_2 = p_{110} = 6/85 \qquad \pi_5 = p_{101} = 12/85$$

$$\pi_3 = p_{200} = 4/85 \qquad \pi_6 = p_{002} = 36/85 \quad .$$

For a large number of states, such as would result for a more reasonable number of messages circulating than the two assumed, solving the set of linear equations given by (2.11) would become costly, since matrix inversion is required.

The alternate method investigated involves three steps: (1) use of the Liu Reduction Method to find the mean first passage time between states, (2) determination of the mean recurrence times for a subset of states, and (3) determination of the state probabilities.

Carrying out step (1) for discrete-time Markov chains requires converting the non-weighted chain (for example, the chain of Figure 2.4.7), to a weighted one. For continuous-time

Markov chains, the extension of Liu's work discussed in the Appendix gives the necessary cost function for state i as $1/q_i$, where $q_i = -q_{ii}$, and q_{ii} is the ith diagonal element of the Q matrix. The appropriate transition probabilities from state i to state j for the cost-weighted chain are:

$$\{q_{ij}/q_i\} \qquad i \neq j \ .$$

The chain with the appropriate cost function is shown in Figure 2.4.8. To continue with the example, the cost-weighted chain of Figure 2.4.8 can be reduced to two states using the Reduction Method. Steps in the reduction procedure are shown in Figures 2.4.9a - 2.4.9f; Figure 2.4.9g gives the final result.

It is a property of the cost-weighted chains constructed in the reduction method that at any stage of the reduction process the cost of a state is equal to the mean first passage time from that state to the group of states remaining at that stage of the reduction process. Thus from Figure 2.4.9g, the mean first passage time from state 2 to state 3 is 27/4, while that from state 3 to state 2 is 1/3.

The second step in the alternative method is to calculate mean recurrence times for the states required from the basic relation, [1],

$$q_j M_{jj} = 1 + \sum_{i \neq j} q_{ji} M_{ij} \ , \qquad (2.14)$$

where M_{jj} is the mean recurrence time of state j.

In the example, if π_3 is the only state probability required, it can be computed from M_{33}. The mean recurrence time M_{33} is computed as follows

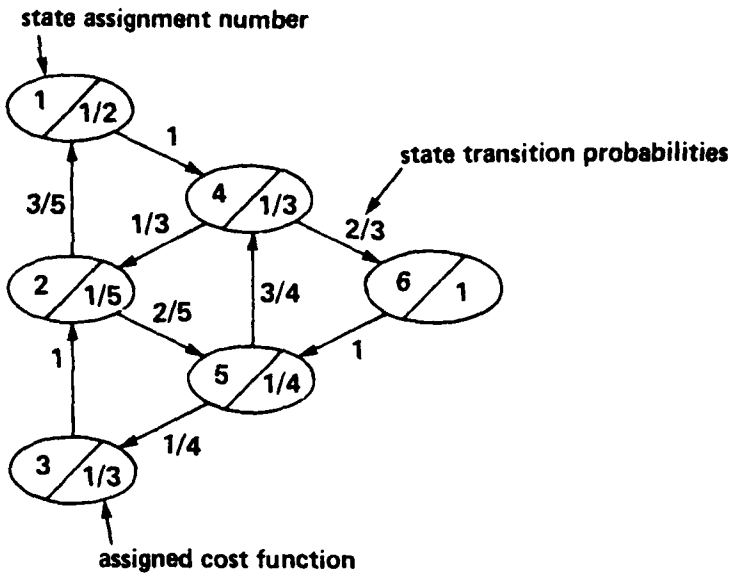


FIGURE 2.4.8. COST–WEIGHTED MARKOV CHAIN.

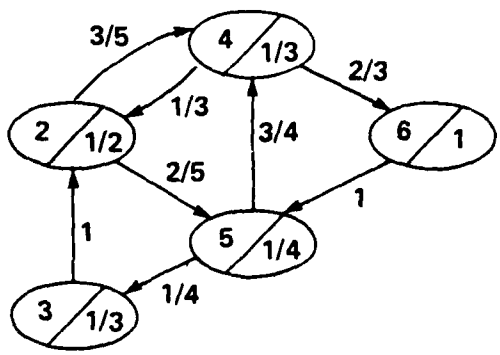


FIGURE 2.4.9.a

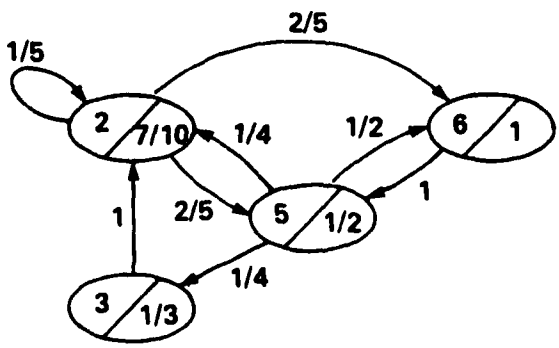


FIGURE 2.4.9.b

FIGURE 2.4.9c

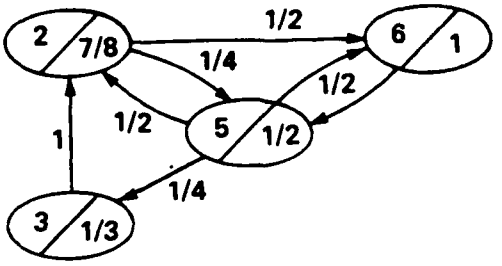


FIGURE 2.4.9d

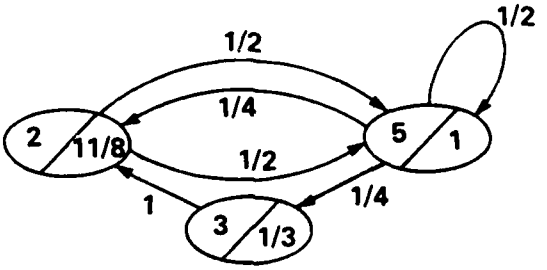


FIGURE 2.4.9e

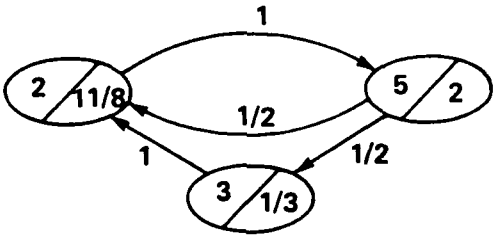


FIGURE 2.4.9f

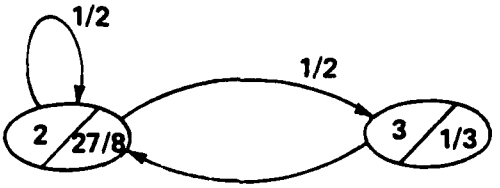
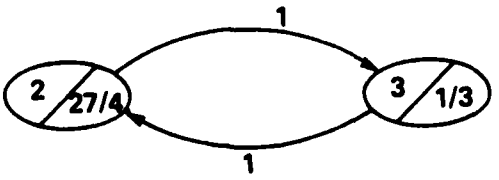


FIGURE 2.4.9g



$$q_3 M_{33} = 1 + q_{33} M_{23}$$
$$3\, M_{33} = 1 + 3 \cdot 27/4 = 85/4$$
$$M_{33} = 85/12 \quad .$$

The third and last step of the alternative method evaluates π_3 from the basic relation

$$\pi_i = \frac{1}{q_i M_{ii}} \qquad (2.15)$$

which, for the example gives

$$\pi_3 = 1/q_3 M_{33} = 4/85 \quad .$$

It should be noted that, in general, (2.14) could require a number of M_{ij} for computing M_{jj}. In such a case additional steps would be required in the reduction procedure.

A special case of some interest is the class of networks for which "product-form" solutions apply [8]. The example under consideration, being a closed network, is in this class. Product-form solutions express all state probabilities in terms of the same parameters.

For example, the probability of any state, identified as $\{n_1\ n_2\ n_3\}$, is given by

$$P\left\{\text{state } n_1 n_2 n_3\right\} = G(N) \prod_{i=1}^{3} X_i^{n_i} \qquad (2.16)$$

where

$$N = n_1+n_2+n_3 = 2 \quad ;$$

and the X_i are solutions of

$$\begin{aligned} \mu_1 X_1 &= \mu_2 X_2 \\ \mu_2 X_2 &= \mu_3 X_3 \end{aligned} \quad , \qquad (2.17)$$

and G(N) is a normalization constant. Equation (2.16) can be used to express all X_i in terms of one chosen X_j, such as X_2. For the example, this gives

$$P\left\{\text{state } n_1 n_2 n_3\right\} = G(2)\ \left(\tfrac{2}{3} X_2\right)^{n_1}\ (X_2)^{n_2}\ (2\ X_2)^{n_3} = G(2) X_2^{\,2} (2/3)^{n_1} 2^{n_3} \qquad (2.18)$$

Knowledge of one state probability is sufficient to determine $G(2)\ X_2^{\,2}$ and hence, all state probabilities. To continue with the example above, the state reduction method yields:

$$\pi_3 = P\left\{\text{state } 200\right\} = 4/85.$$

Use of this value gives

$$G(2)X_2^2 = 9/85$$

and

$$P\{\text{state } n_1 n_2 n_3\} = \frac{9}{85}\left(\frac{2}{3}\right)^{n_1}(2)^{n_3} \quad .$$

It should be pointed out that in addition to the method developed, other computational algorithms are discussed in the literature for direct calculation of important system parameters for product-form queueing networks, [4]. Specific mention should be made of the convolutional algorithm of Buzen and the mean value analysis algorithm proposed by Reiser and Lavenberg.

2.5 Discussion

Modeling and mathematical analysis of a computer-communication network must, to some extent, be tailored to particular applications and desired results. This Section reviews the approaches which can be used.

Queueing models are almost always used for performance analysis. Section 2.1 gives references to a number of papers covering background material. Section 2.2 gives specific queueing models for the two major components of the AIRMICS/Georgia Tech network.

Operating conditions for the queueing networks are discussed in Section 2.3 under the heading of "Parameters for the Queueing Model."

Approaches to studies with queueing models are reviewed in Section 2.4. It is pointed out that both analytic and simulation studies of queueing models can be made.

Generally speaking, simulation studies are necessary for fairly accurate results if the network has tandem queues and if the "product-form" assumptions are not justified. Such studies are, however, expensive in computer time and do not give explicit relations between variables. Simulation studies would seem to be appropriate when all but a few parameters are specified, (including input message distributions and message length distributions), and accurate curves are desired for the relationships between, at most, several variables.

Analytic studies tend to be approximate but have the advantage of yielding explicit relations between the variables. The report gives conditions such that a very tractable product-form solution can be obtained.

One restriction on the product-form solution is the assumption of infinite storage at the nodes. This assumption is obviously not valid and its removal is necessary in order to study certain important effects. The report reviews work on this problem from the literature and also gives the results of a new method developed on the project.

With respect to analytical studies, it is recommended that first cut approximate solutions be obtained from assumptions yielding the most tractable product-form solutions. Such solutions are used in Section 3 in analyzing controlled experiments with the network. In spite of their approximate nature, there is considerable evidence for adequate accuracy from the very tractable models.

If blocking and other problems asociated with finite buffers are to be studied, the choice of approach is less clear. All of the methods reviewed, including the one developed on the project, have restrictions. For issues of buffer management and flow control at a single node, the methods of Irland, Lam and Reiser seem to be attractive.

On the other hand, for complete networks the approximate method of Penotti and Schwartz seems to be the most tractable available. Their method is not general and its restrictions have not been investigated by the authors of this report.

The most accurate, method of working with finite storage at the network nodes uses a Markov chain type model. This type of model is accurate, but typical parameter values lead to an extremely large number of states in the model. Conventional approaches require the inversion of very large matrixes. The method developed on this project attacks the problem of large numbers of states by a state reduction technique. The method could have advantages when it is necessary to model more than one node of the network in considering blocking, deadlock or some other type of degradation. The use of

mean first passage time from some reference state to some other state, or group of states where some type of degradation occurs can act as a criterion for network design.

2.6 References for Section 2

1. Kleinrock, L., Queueing Systems, Vol. 1 - Theory. New York: Wiley-Interscience, 1975.

2. Kobayashi, H. and Konheim, A., "Queueing Models for Computer Communications System Analysis", IEEE Trans. Comm., Vol. COM-25, No. 1, January 1977, pp. 2-28.

3. Cohen, J. W., "The Single-Server Queue", North Holland: Amsterdam, 1969.

4. Wong, J., "Queueing Network Modeling of Computer Communication Networks", Computing Surveys, Vol. 10, No. 3, September 1978, pp. 343-350.

5. Iobagi, F., et al, "Modeling and Measurement Techniques in Packet Communication Networks", Proc. IEEE, Vol. 77, No. 11, November 1978, pp. 1423-1447.

6. Samari, N. and Schneider, G., "A Queueing Theory-Based Analytical Model of a Distributed Computer Network", IEEE Trans. on Computers, Vol. C-29, No. 11, November 1980, pp. 994-1001.

7. Kleinrock, L. and Naylor, W., "On Measured Behavior of the ARPA Network", in Proc. 1974 AFIPS National Computer Conference, Vol. 43, pp. 767-780.

8. Kleinrock, L., Queueing Systems Vol. 2: Computer Applications, New York: John Wiley and Sons, 1976.

9. Labetoulle, J., E. G. Manning and R. W. Peebles, "A Homogeneous Computer Network", Computer Networks, Vol. 1, 1977, pp. 225-240.

10. Kritzinger, P., A. Krzesinski, and P. Teunissen, "Incorporating System Overhead in Queueing Network Models", IEEE Trans. on Software Engineering, Vol. SE-6, No. 4, July 1980, pp. 381-390.

11. Ibid. [8], pp. 320-325.

12. Baskett, F. etal, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", JACM, Vol. 22, No. 2, April 1975, pp. 248-260.

13. Spragins, J., "Approximate Techniques for Modeling the Performance of Complex Systems", Computer Languages, Vol. 4, 1979, pp. 99-129.

14. Irland, M., "Buffer Management in a Packet Switch", IEEE Trans. on Comm., Vol. COM-26, No. 3, March 1978.

15. Lam, S. S., "Store-and-Forward Buffer Requirements in a Packet Switching Network", IEEE Trans. on Comm., Vol. COM-26, No. 4, April 1976.

16. Lam, S. S., and M. Reiser, "Congestion Control of Store-and-Forward Networks by Input Buffer Limits - An Analysis", IEEE Trans. on Comm., Vol. COM-27, No. 1, January 1979.

17. Penotti, M. C. and M. Schwartz, "Congestion Control in Store-and-Forward Tandem Links", IEEE Trans. on Comm., Vol. COM-23, No. 12, December 1975.

18. Rudin, H., "An Introduction to Flow Control", Proceedings ICCC, Toronto, Canada, August 1976, pp. 463-466.

19. Kaufman, L., G. Gopinath and E. F. Wunderlich, "Analysis of Packet Network Congestion Control Using Sparse Matrix Algorithms", IEEE Trans. on Comm., Vol. COM-29, No. 4, April 1981, p. 453-465.

20. Liu, S. S., "Analysis of Reframing Performance of Multilevel Synchronous Time Division Multiplex Hierarchy", Ph.D. Thesis, Georgia Institute of Technology, 1979.

21. Chandy, K. M. and C. H. Sauer, "Computational Algorithms for Product Form Queueing Networks", CACM, Vol. 23, No. 11, October 1980.

22. Chung, K. L. "Markov Chains with Stationary Transition Probabilities", Heidelberg: Springer-Verlag, 1967.

23. Cinlar, E., "Introduction to Stochastic Processes", Prentice-Hall, Englewood Cliffs, NJ, 1954.

24. Gelenbe, E. and J. Mitrani, "Analysis and Synthesis of Computer Systems", London: Academic Press, 1980.

APPENDIX

For discrete-time irreducible recurrent Markov chains, it is fairly straightforward [3] to show that the mean first passage times $\{M_{jk}, j \neq k\}$ are related by the set of linear equations:

$$M_{ik} = 1 + \sum_{j \neq i} p_{ij} M_{jk} \quad \text{for all } j \neq k \tag{A.1}$$

where $\{P_{ij}\}$ are the transition probabilities of the Markov chain. The mean recurrence time for any state k is likewise given by:

$$M_{kk} = 1 + \sum_{j \neq k} p_{kj} M_{jk} \quad . \tag{A.2}$$

A similar set of equations holds for continuous-time irreducible recurrent Markov chains, namely

$$q_i M_{ik} = 1 + \sum_{j \neq k} q_{ij} M_{jk} \quad \text{for all } i \neq k \tag{A.3}$$

and

$$q_k M_{kk} = 1 + \sum_{j \neq k} q_{kj} M_{jk} \quad \text{for all } k$$

where q_{ij}, $i \neq j$ is the transition rate from state i to state j and $q_i = q_{ii}$ is the total transition rate out of state i. As a consequence of the definitions:

$$q_i = \sum_{j \neq i} q_{ij} \tag{A.5}$$

$$\sum_j q_{ij} = 1 \quad \text{all } i. \tag{A.6}$$

The transition rate matrix of the continuous time Markov chain is denoted Q and is given by:

$$Q = [q_{ij}] \qquad i, j = 0, 1, \ldots$$

To prove (A.3) and (A.4) rigorously is quite difficult, some fundamental theorems from renewal theory are involved (see for example, [22] and [23]).

If (A.3) is rewritten as:

$$M_{ik} = 1/q_i + \sum_{j \neq k} \left(\frac{q_{ij}}{q_i}\right) M_{jk} \qquad (A.7)$$

Liu's reduction method can be applied to the continuous-time chain using $1/q_i$ as the cost function for state i and

$$r_{ij} = q_{ij}/q_i \quad , \quad j \neq i$$

as the transition probability from state i to state j. Note that $r_{ij} = 0$ for all i, so that no self-loops are permitted in the cost-weighted chain. Since

$$\sum_j r_{ij} = \sum_{j \neq i} q_{ij}/q_i = \frac{1}{q_i}(-q_{ii}) = 1 \quad ,$$

$\{r_{ij}\}$ is a valid transition probability matrix.

Using Liu's method, M_{jk} , $j \neq k$ can be evaluated for all j for which $q_{kj} > 0$. Then (A.4) yields $q_k M_{kk}$, which can then be used to express the stationary probability of state k as:

$$\Pi_k = 1/q_k M_{kk} \qquad (A.8)$$

Equation (A.8), while quite intuitive, is also proven rigorously using renewal theory [24].

3. CONTROLLED EXPERIMENTS

FOR STUDYING MATHEMATICAL MODELS

3.1 Introduction

Section 2 of this report presents queueing models of the components of the AIRMICS/Georgia Tech network and discusses several sets of assumptions which lead to alternative analytic solutions. The purpose of this section is to describe several experiments to be made with elementary network configurations under controlled conditions. These experiments are tailored to provide tests of the various mathematical models for assessing their accuracy.

The experiments are regarded as a first step in evaluating the models under conditions which attempt to focus on one or two effects at a time. Results of the experiments are expected to lead to refinements of the models and possibly refinements in the physical network. It is anticipated that later work will be directed toward studies of more typical network configurations under realistic conditions using both the models and the physical network.

The material in this section will be presented in a format which describes an experiment, gives a queueing model of the network configuration, and finally gives an analytic solution for measurable quantities, such as average message delay, as a function of network parameters.

Several Appendices to this section present details of the analytic solutions which were not covered in Section 2.

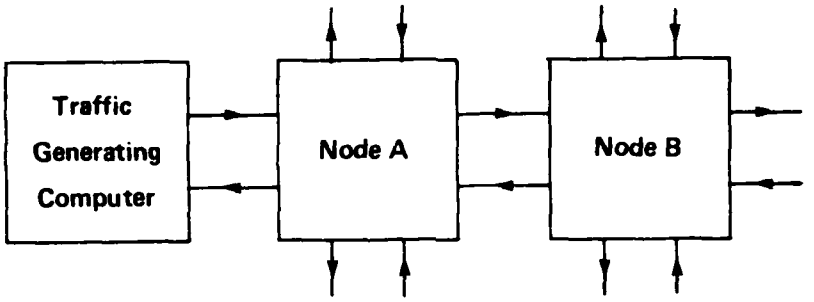
3.2 Experimental Network Configurations

The proposed experiments configure the network in two ways: as a loop of two nodes and as a loop of three nodes as shown in Figure 3.2.1. For both configurations traffic is generated by a traffic generating computer and for the experiments each message is one packet in length.

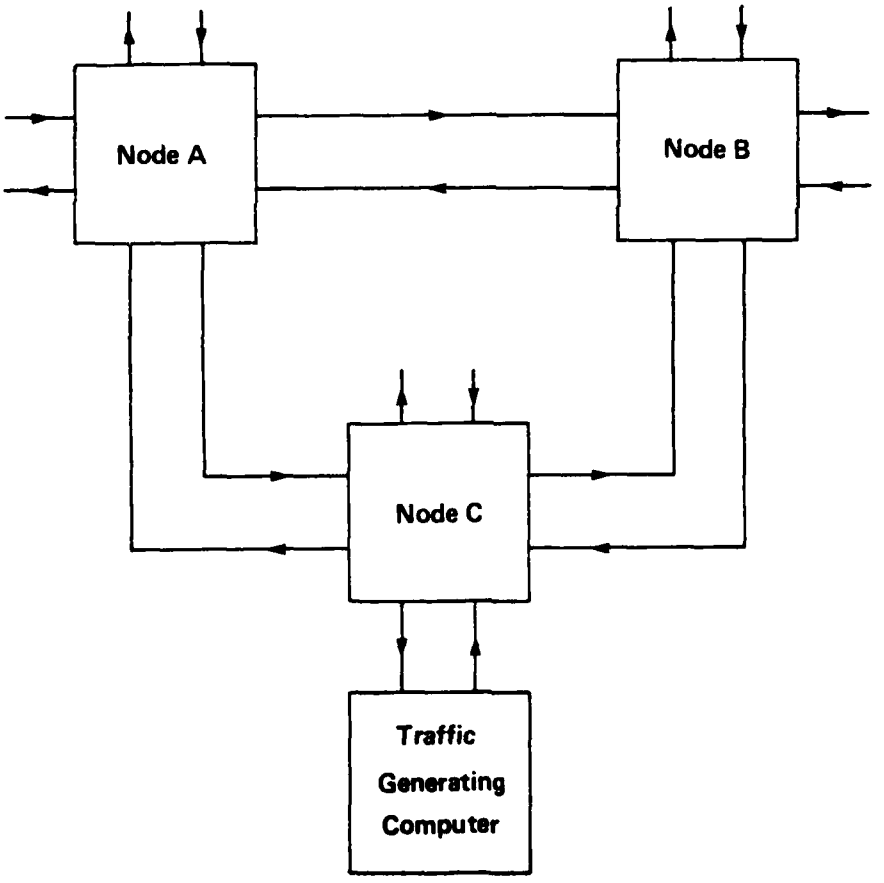
Two experiments involve "open networks" in the sense that messages arrive at the network and then pass through it without returning.

Two experiments involve "closed networks" for which a fixed number of messages circulate around the network. In practice this could be accomplished by connecting the output terminals of the network back to the input terminals with a fixed number of messages stored in the network buffers prior to actuating the system. For the experiments, it is more convenient to achieve a closed network by programming the message generating computer to input a new message to the network each time a message exits from the network.

In addition to the four experiments described above, a fifth experiment is designed to investigate the effects of finite storage using an open configuration.



(a) TWO–NODE CONFIGURATION



(b) THREE–NODE CONFIGURATION

FIGURE 3.2.1. EXPERIMENTAL NETWORK CONFIGURATIONS.

The two experimental configurations are chosen for the following reasons. The network is designed so that positive acknowledgements are generated when a packet is correctly received in transmission between nodes. Thus, for example, if a packet is transmitted from Node A to Node B and received correctly, an acknowledgement packet is sent from Node B back to Node A.

The two-node configuration chosen is the simplest configuration for which acknowledgement traffic combines and queues with data traffic for use of the links. This is apparent by considering Figure 3.2.1a. Data traffic from Node A to Node B traverses the top member of the full duplex link. Upon correct receipt at Node B, the data packets are "turned around" with negligible delay and transmitted from Node B to Node A. But acknowledgements are generated for each correctly received packet and the acknowledgements are also sent from Node B to Node A. Similar reasoning shows that acknowledgement packets combine with data packets in going from Node A to Node B.

The three-node configuration was chosen as a configuration for which acknowledgement packets do not combine with data packets. Examination of Figure 3.2.1b shows that for the three-node configuration, acknowledgement packets, for example, will flow over the lower link from Node B to Node A while the data packets, which generated the acknowledgements, flow from Node B to Node C over the lower link. Similar consideration of the other links shows that data and acknowledgement packets remain separate on these links also.

The traffic generating computer can control several features of the generated traffic, namely:

1. The packet length distribution
2. The mean packet length, $1/\mu$
3. The packet arrival rate, λ .
4. The distribution of the interarrival times to the first node.

Exponential length distributions are required for exact adherence to the requirements for the tractable M/M/1 "independant" analytical model. Fixed length packets are possibly easier to generate and, in many cases, provide results which can be adequately approximated by the tractable models. Both types of packets are used in the experiments. Mean packet length and packet arrival rate are parameters in the experiments described.

For the experiments, the measured quantities are different delay times, all of which are measured with respect to the first bit in the packet. Total average message delay is measured as the average time required for packets to travel from the traffic generator through the network and back to the traffic generating computer. The average is taken over a large number of packets. Nodal delay is measured as the average transit time for packets from entry into a particular node to arrival at the next destination node.

Several assumptions are made in obtaining the analytical models for the experiments. First the "independence" assumption is used and this requires that packet lengths be selected independently from an

exponential distribution at each node. This assumption can be satisfied at the first node, where messages enter, but is only an approximation at other nodes.

Infinite buffers will be assumed. This is an assumption for open networks that is approximately true if the average number of packets queueing at a given node is considerably less than the finite buffer size. For closed networks the approximation is valid if the number of messages circulating is less than the finite buffer capacity at each node.

Figure 2.1 of Section 2 gives a relatively complete queueing network for a switching node. The processing time for the CPU to transfer packets out of the input line buffer is short compared to the time for transferring packets out of the nodal buffer over the line to an adjacent node. Thus, it can be assumed that input line buffers can be neglected, as such, and their effect included with nodal processing time. Using this assumption, each node can be represented by a single queue for each output line in a queue model.

Finally, in general, transmission over a line has an associated propagation delay which is typically on the order of tens of milliseconds per thousand miles. Clearly such delays are negligible for the very short times used in the experiments.

3.3 Experiment One

This Experiment uses the three node configuration of Figure 3.2.1b. The traffic generator is set up for operation as an open network with packet length and packet arrival rate as parameters. Average total message delay and average nodal delays are measured.

A queueing model for the network configured for this experiment (and using the assumptions stated) is given in Figure 3.3.1. The average nodal delay, T_i, for a typical node, assuming exponentially distributed message lengths, can be expressed as

$$T_i = \frac{1}{\mu C - \lambda} + K ,$$

Where K is the nodal processing time for the particular node. The total average message delay, T, through the whole network can be obtained by summing the four nodal delays to obtain

$$T = \frac{4}{\mu C - \lambda} + K_A + K_B + 2K_C .$$

For purposes of comparison, the average nodal delay, assuming a constant packet length, is given by

$$T_i = \frac{1 - \lambda/2\mu C}{\mu C - \lambda} + K .$$

This result is derived, for example, by Kleinrock, [1].

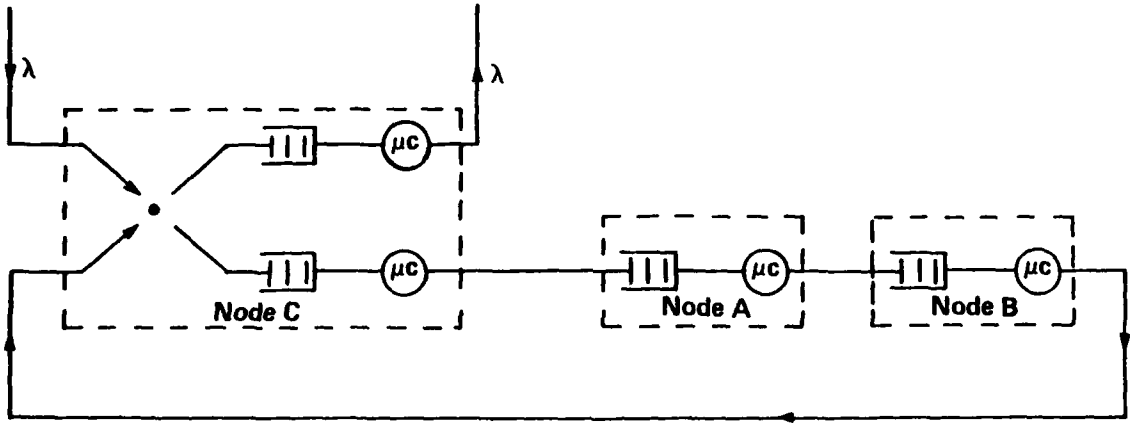


FIGURE 3.3.1. A QUEUEING MODEL FOR THE NETWORK CONFIGURATION USED IN EXPERIMENT ONE.

In presenting curves for total average message delay and average nodal message delay, it is convenient to normalize the expressions given above to obtain

nodal delay: $(T_i - K) = \frac{1/\mu C}{1-\rho}$

total delay: $(T - K_A - K_B - 2K_C) = \frac{4/\mu C}{1-\rho}$

nodal delay, constant message length: $(T_i - K) = \frac{[1-\rho/2]}{1-\rho}(\frac{1}{\mu C})$

where

$$K = K_A \text{ or } K_B \text{ or } K_C$$

$$\rho = \lambda/\mu C \quad .$$

The normalized average nodal delay curves for both exponentially distributed and constant message lengths are given in Figure 3.3.2, while the normalized average total delay curve is given in Figure 3.3.3.

For reference, unnormalized curves for average total message delay are given in Figures 3.3.4 and 3.3.5 for C = 1200 bits/sec., K_A, K_B, and K_C assumed to be zero, and message lengths of 500 and 2000 bits. A curve for average nodal delay with both fixed and exponentially distributed message lengths is given in Figure 3.3.6 for C=1200 bits/sec., K assumed to be zero and an average message length of 500 bits.

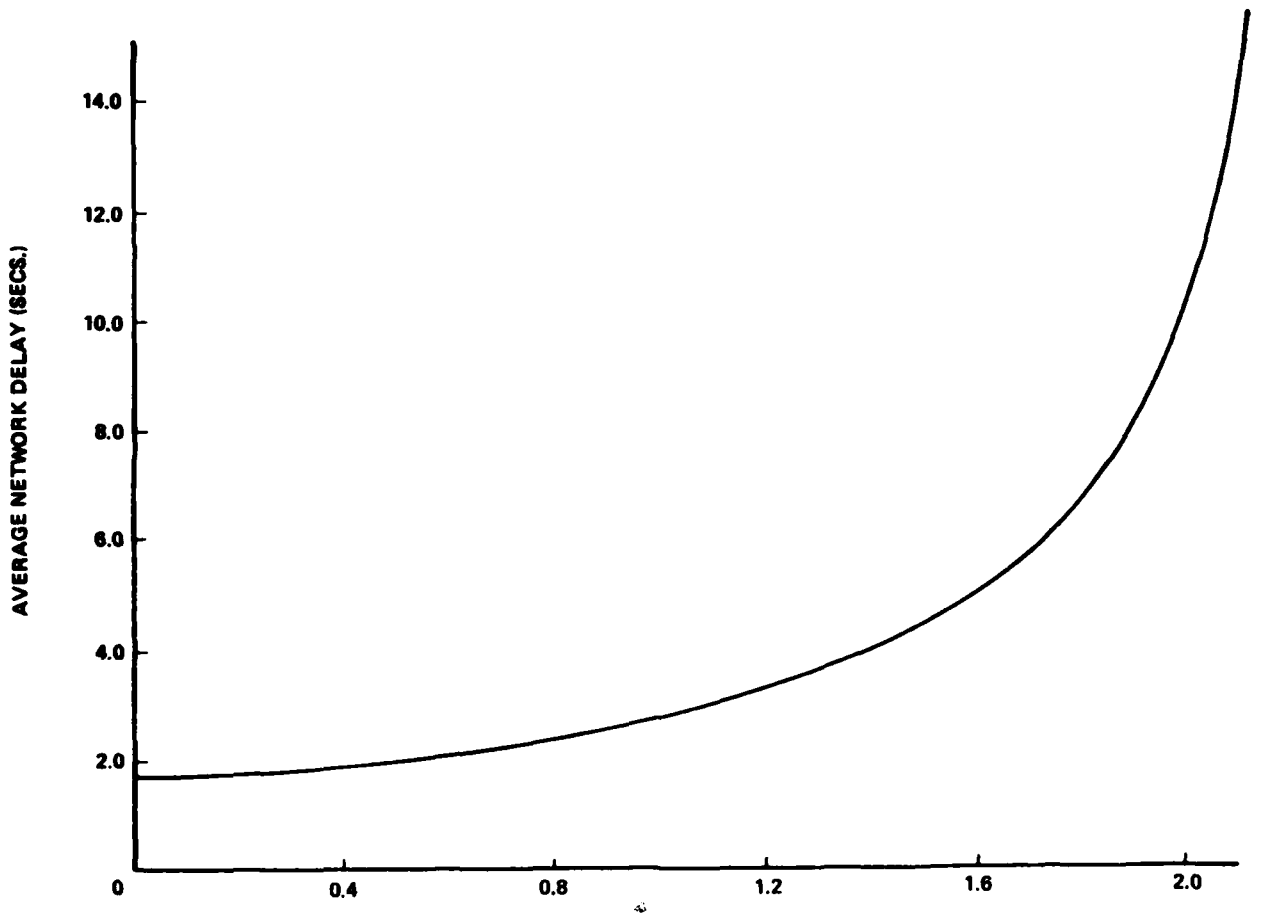


FIGURE 3.3.4. AVERAGE MESSAGE DELAY VERSUS MESSAGE/SEC. FOR MESSAGE LENGTH = 500 BITS, C = 1200 BITS/SEC., $K = K_E = 0$.

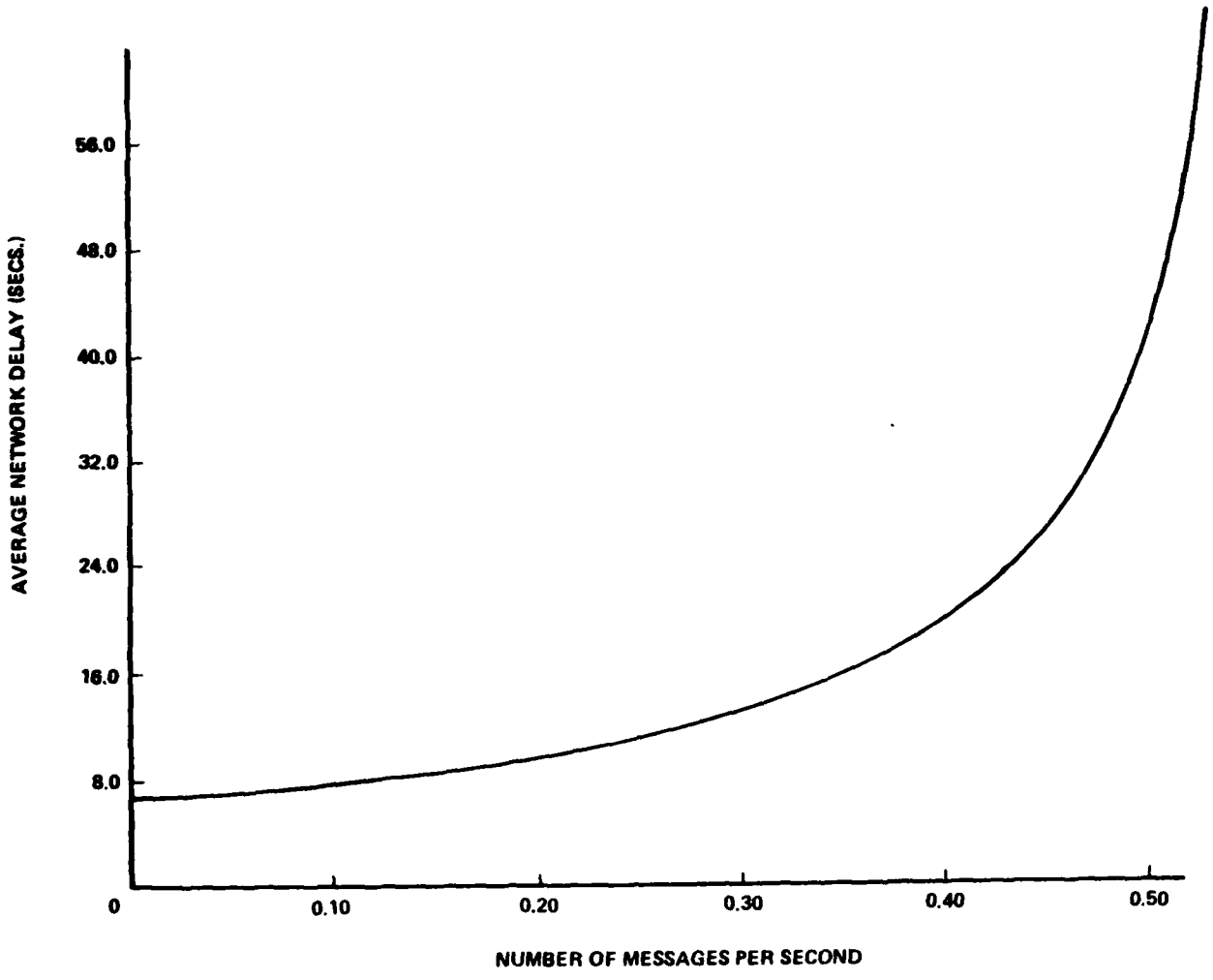


FIGURE 3.3.5. AVERAGE MESSAGE DELAY VERSUS MESSAGES/SEC. FOR MESSAGE LENGTH = 2000 BITS, C = 1200 BITS/SEC., $K = K_E = 0$.

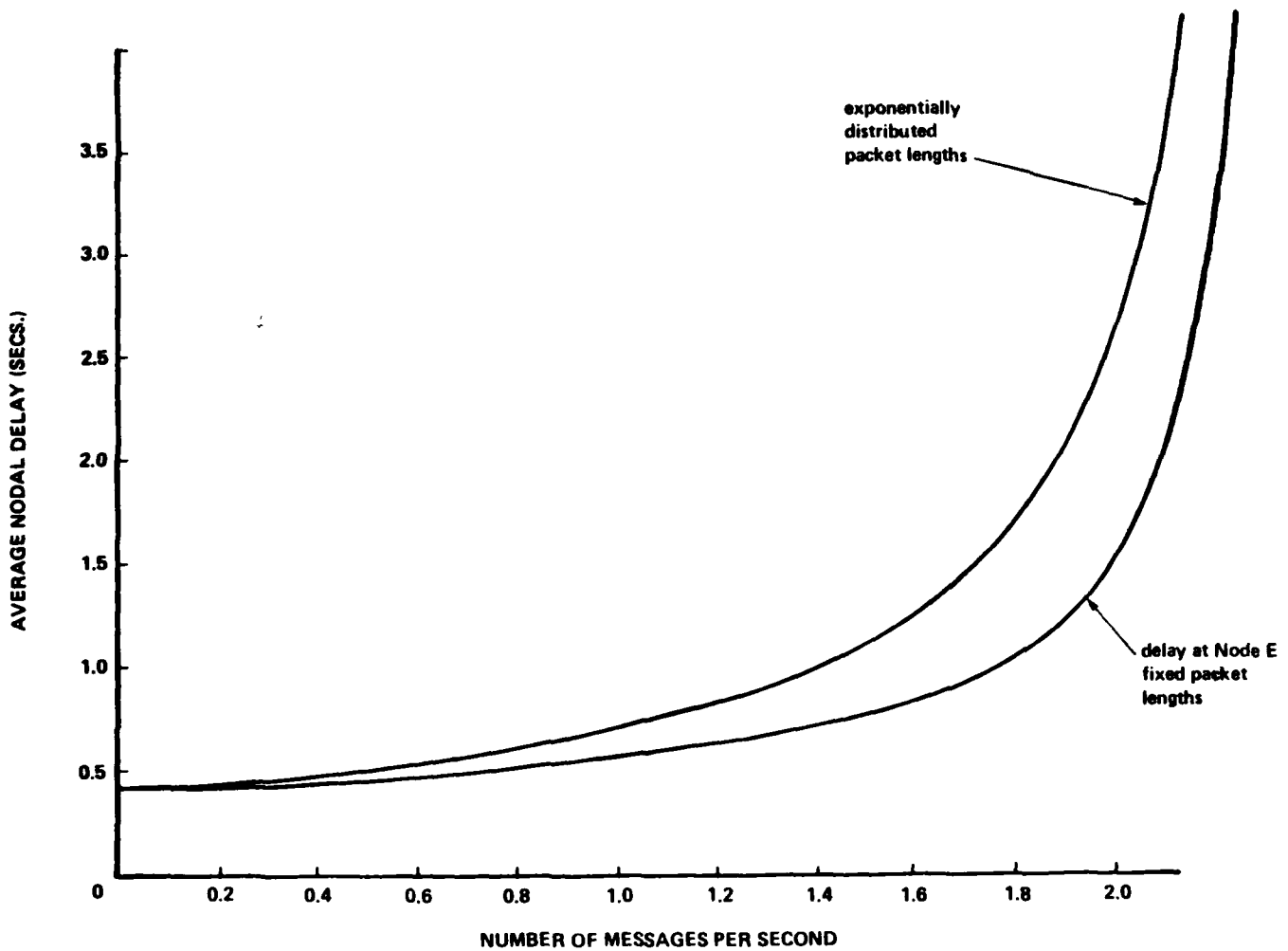


FIGURE 3.3.6. AVERAGE NODAL DELAY VERSUS MESSAGES/SEC. FOR *MESSAGE LENGTH = 500 BITS, C = 1200 BITS/SEC.*, $K = K_E = 0$.

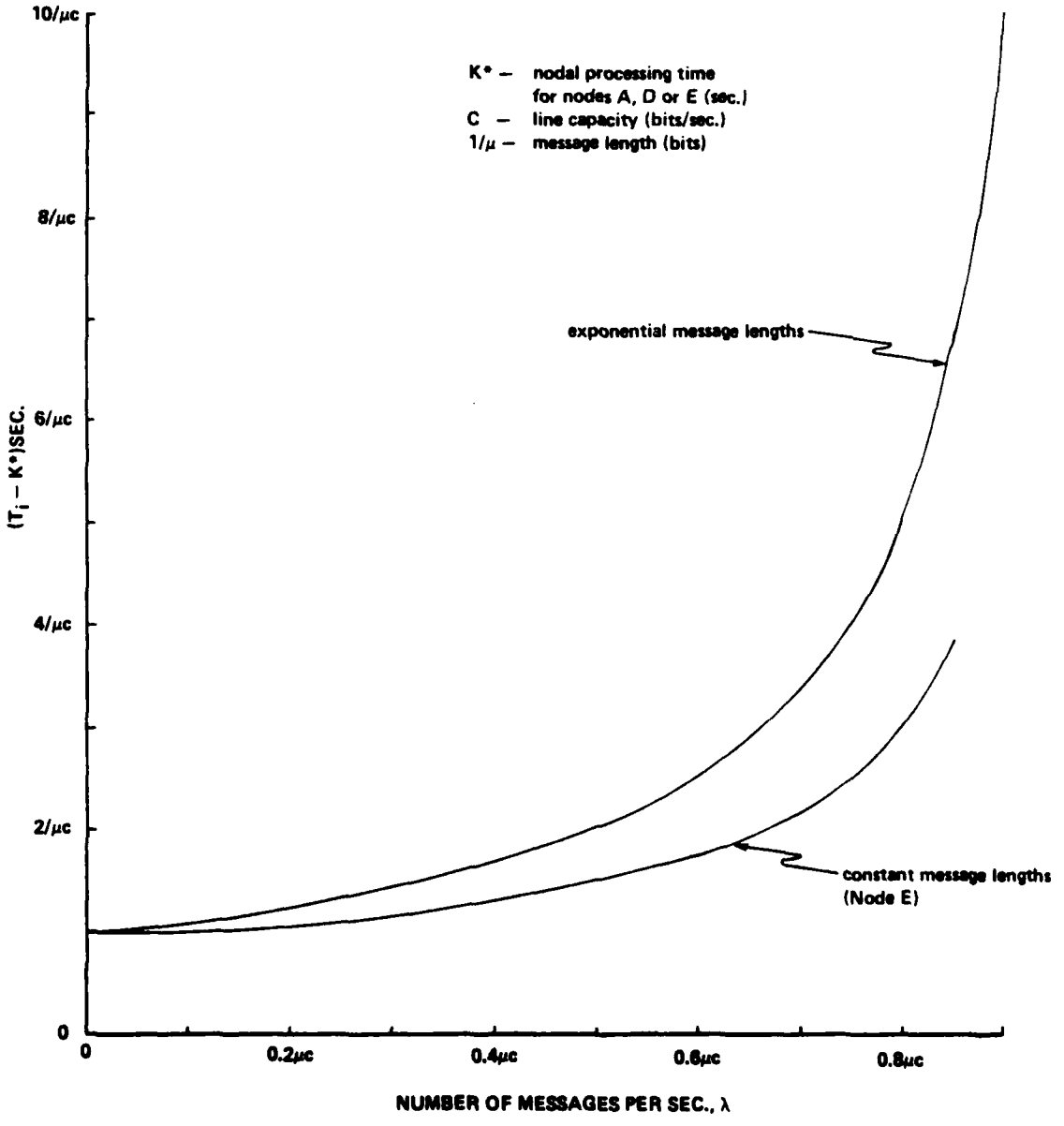


FIGURE 3.3.2. AVERAGE NODAL DELAY (NORMALIZED UNITS) VERSUS MESSAGES/SEC. (NORMALIZED UNITS).

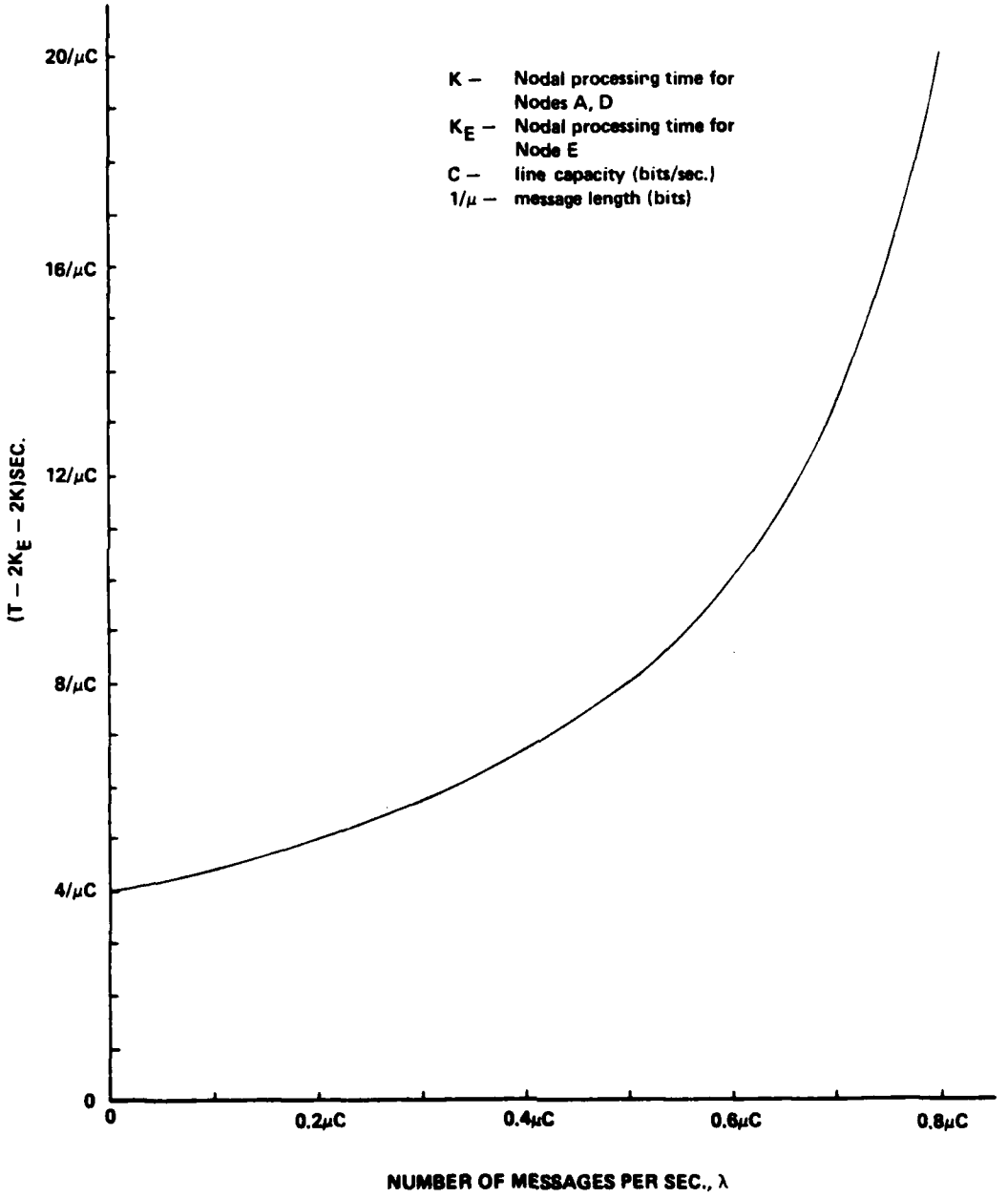


FIGURE 3.3.3. TOTAL AVERAGE MESSAGE DELAY (NORMALIZED UNITS) VERSUS MESSAGES/SEC. (NORMALIZED UNITS).

3.4 Experiment Two

This experiment configures the network into the two-node configuration shown in Figure 3.2.1a. As with Experiment One, this Experiment is with an open network, and traffic generation conforms with this requirement. Packet length and packet arrival rate are parameters and average total message delay and average nodal delay are the measured quantities.

A queueing model for the configuration is given in Figure 3.4.1. Note that the traffic labeled in the Figure accounts for acknowledgements as well as data traffic. It is assumed that the combined data and acknowledgement traffic still forms a Poisson process. In general, the acknowledgement packets have different lengths from the data packets and nodal delays for combined data and acknowledgement traffic are determined by an average packet length $1/\mu'$, given by

$$1/\mu' = \frac{1}{2}\,(1/\mu + 1/\mu_{ack}) \quad .$$

The total average message delay from the output of the message generator back to its input, subject to the present assumptions, is given by

$$T = \frac{1}{\mu C-\lambda} + \frac{4\lambda/\mu' C}{\mu' C-2\lambda} + {}^{2}/\mu C + 2K_A + K_B \quad .$$

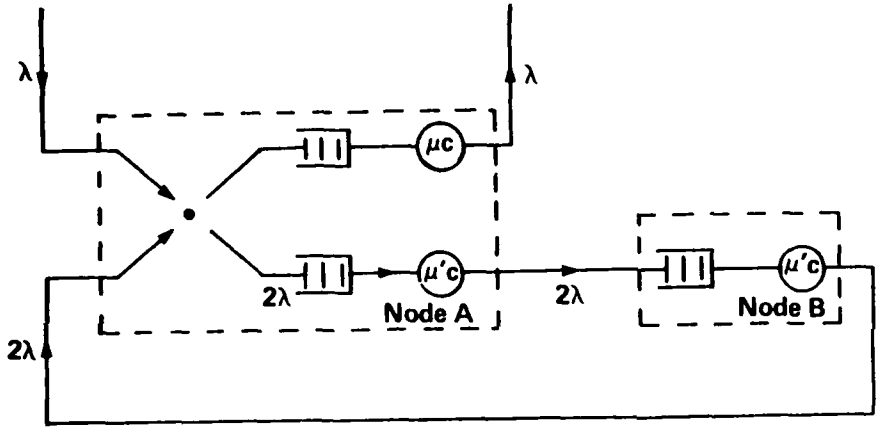


FIGURE 3.4.1. A QUEUEING MODEL FOR THE NETWORK CONFIGURATION USED IN EXPERIMENT TWO.

Unfortunately, this total average message delay cannot be normalized in a useful manner, as was the case for Experiment One, and so results are plotted for several typical cases. Additional curves can be obtained easily for other sets of parameter values.

Values chosen for the curves are as follows:

link capacity = 1200 bits/sec.

$1/\mu_{ack}$ = 200 bits

$1/\mu$ = 500, 1000, 1500, 2000 bits/sec.

$K_A = K_B = 0$.

Figures 3.4.2 through 3.4.5 give average message delay versus message arrival rate (throughput) for the sequence of message lengths listed. The curves also show average message delay for the same conditions without acknowledgement traffic. Note that, as would be expected, the effect of acknowledgements is reduced as message length is increased for fixed acknowledgement packet length.

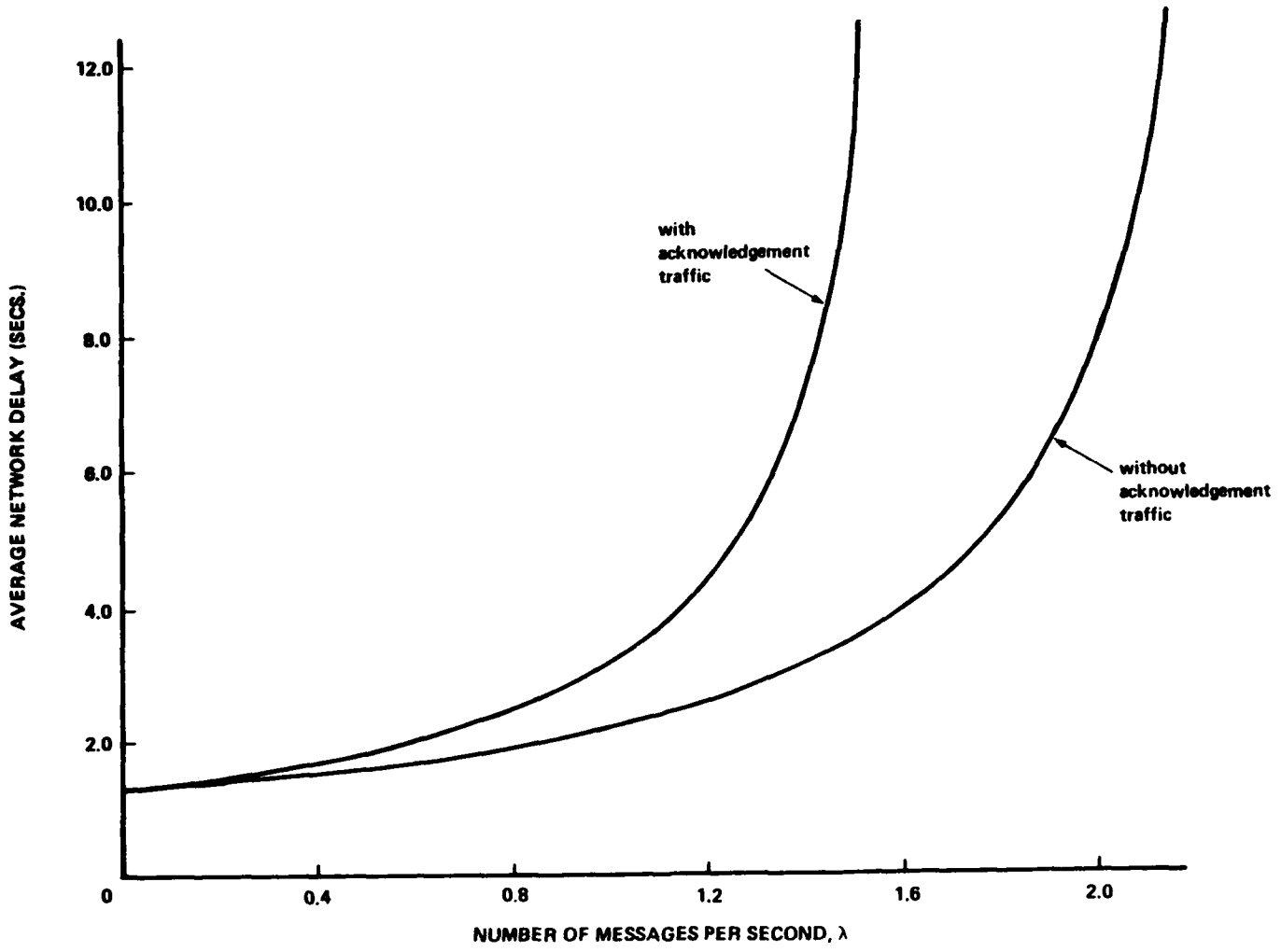


FIGURE 3.4.2. AVERAGE MESSAGE DELAY VERSUS THROUGHPUT FOR MESSAGE LENGTH = 500 BITS, C = 1200 BITS/SEC., $K_A = K_B = 0$.

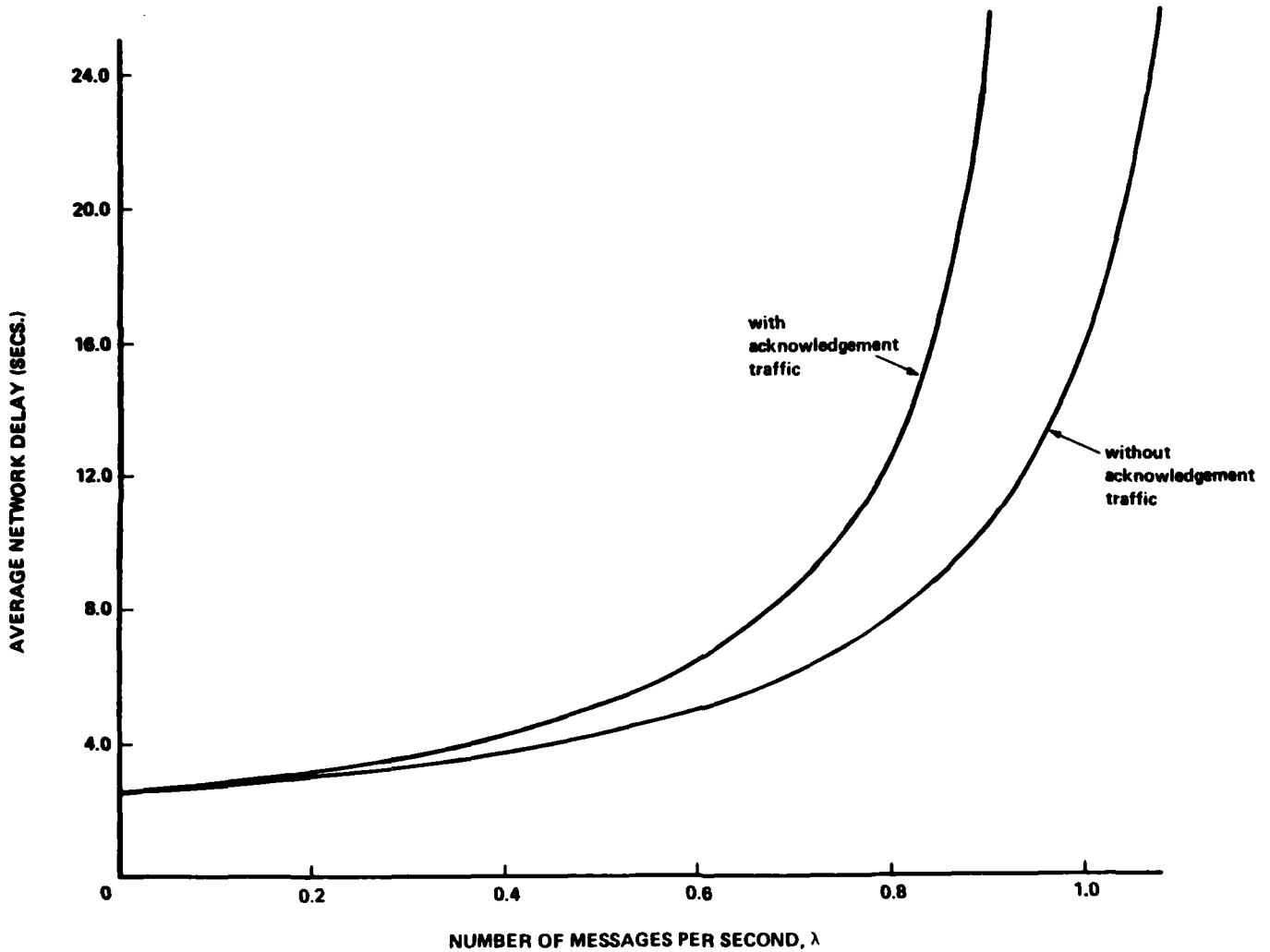


FIGURE 3.4.3. AVERAGE MESSAGE DELAY VERSUS THROUGHPUT FOR MESSAGE LENGTH = 1000 BITS, C = 1200 BITS/SEC., $K_A = K_B = 0$.

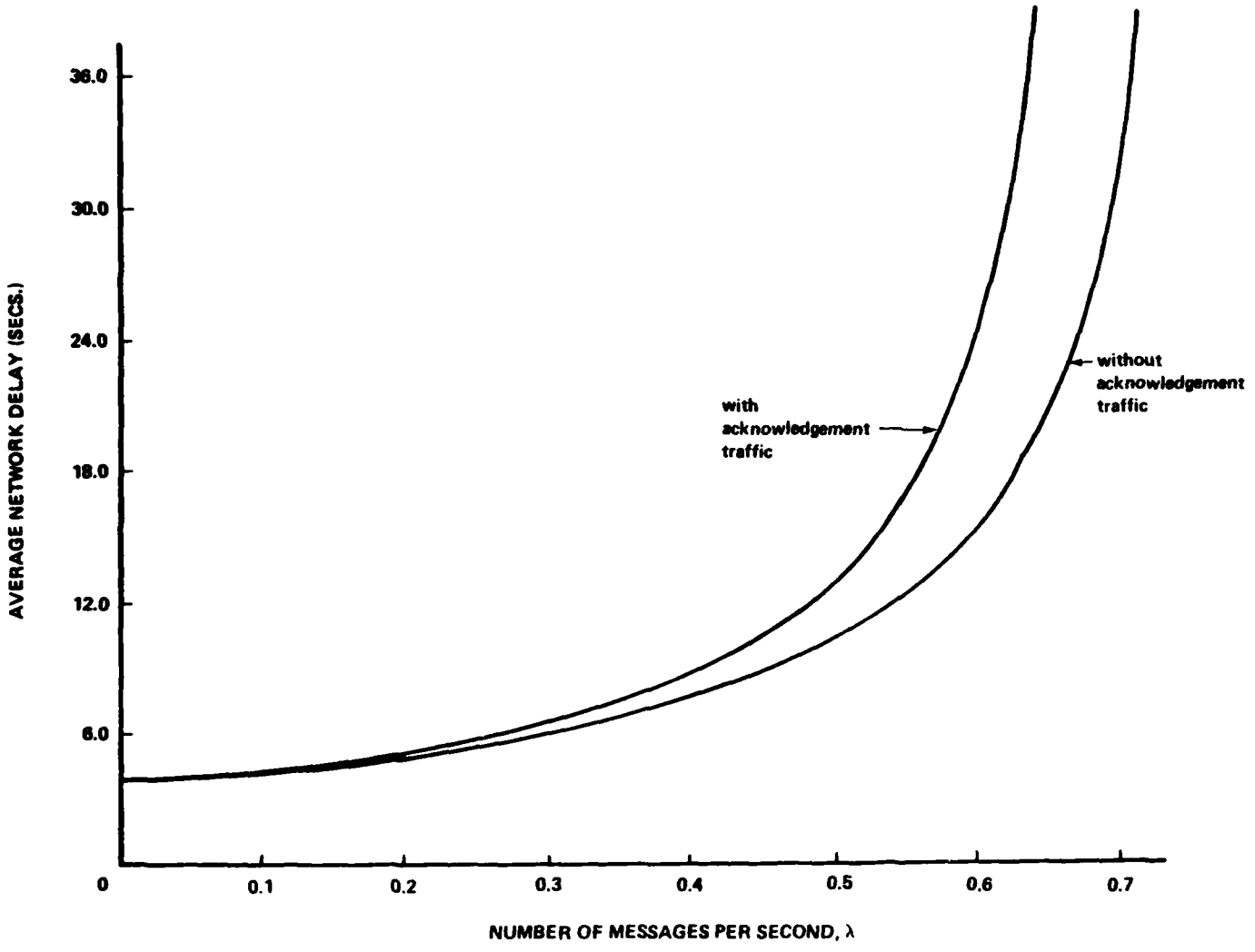


FIGURE 3.4.4. AVERAGE MESSAGE DELAY VERSUS THROUGHPUT FOR MESSAGE LENGTH = 1500 BITS, C = 1200 BITS/SEC., $K_A = K_B = 0$.

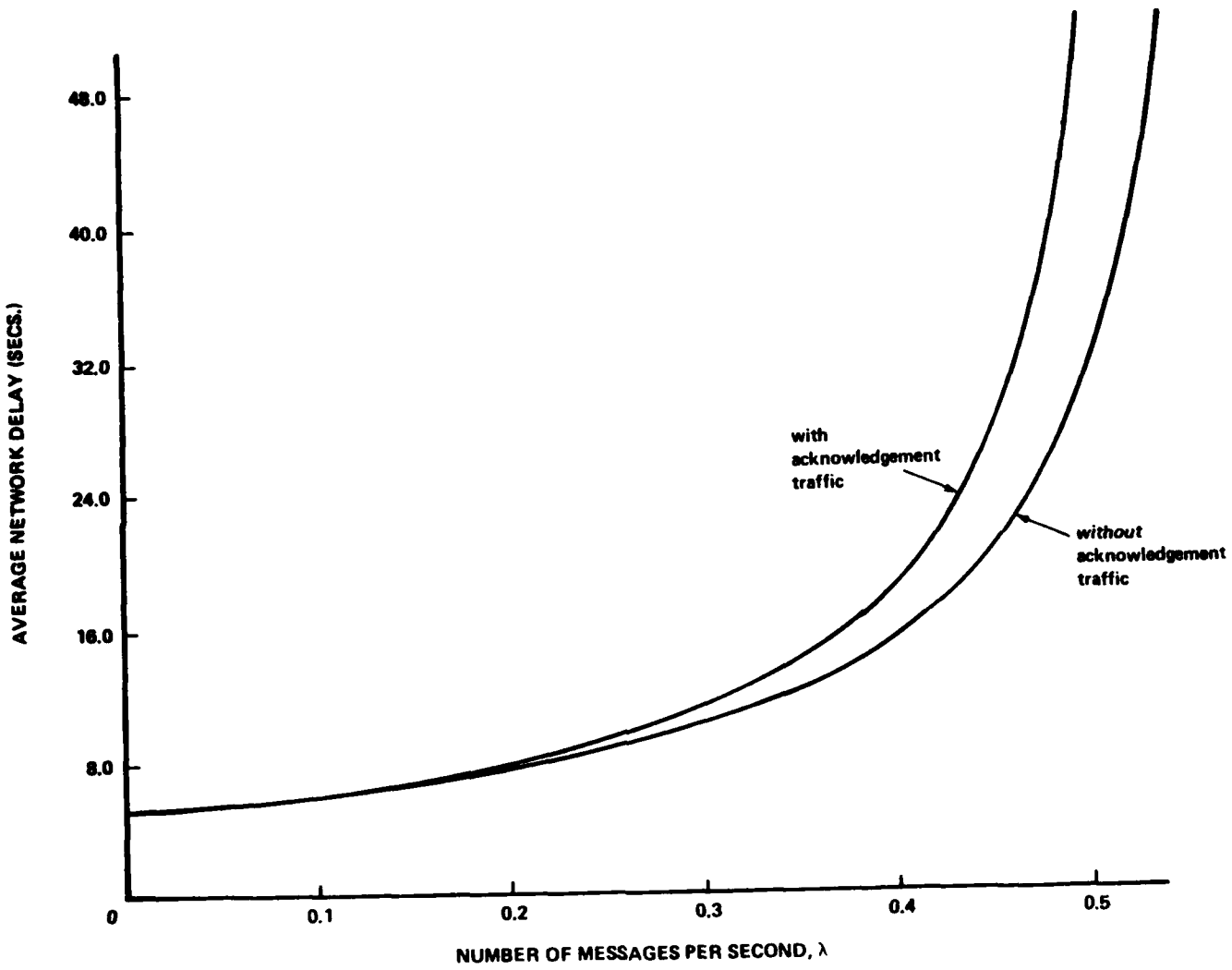


FIGURE 3.4.5. AVERAGE MESSAGE DELAY VERSUS THROUGHPUT FOR MESSAGE LENGTH = 2000 BITS, C = 1200 BITS/SEC., $K_A = K_B = 0$.

3.5 Experiment Three

This Experiment uses the three node configuration of Figure 3.2.1b, but differs from Experiment One in that in this case the network is closed. The traffic generator is set up for closed system operation in that it replaces every exiting packet immediately with a new input packet. The number, N, of messages circulating in the system is a parameter of the Experiment.

Figure 3.5.1 gives a queueing model for this Experiment. The model is analyzed under two sets of assumed conditions. Under the assumption that the message lengths at each node are independent and exponentially distributed, it is possible to obtain an analytic solution for average message delay through the system. The result,

$$T = \frac{N+3}{\mu C} \quad ,$$

is derived in Appendix 3.1.

For assumptions other than exponential distribution of message lengths, it does not seem to be possible to obtain an analytic result for average message delay. However, if the exponential message length assumption is replaced by the assumption that message lengths are fixed and non-random, which is a better approximation to the real system, it is possible to obtain useful lower bounds on average message delay.

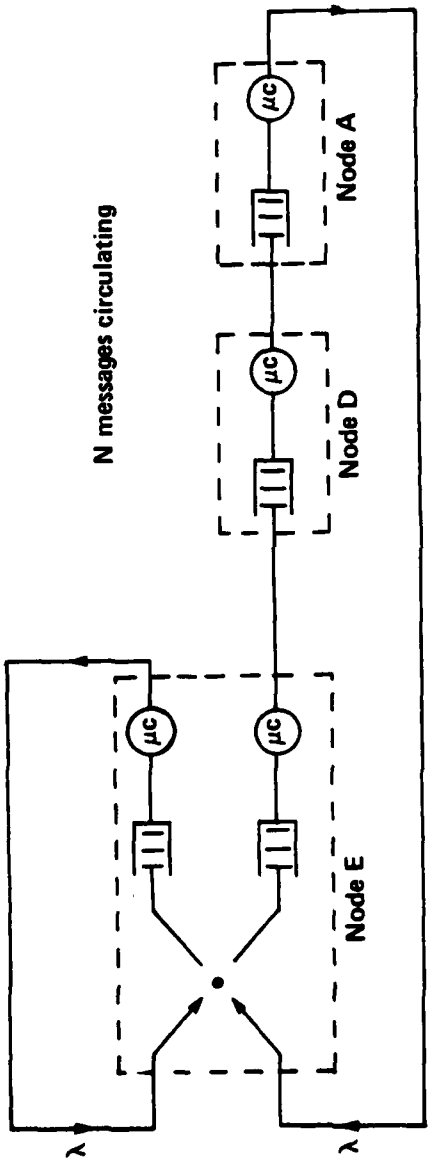


FIGURE 3.5.1. QUEUEING MODEL OF CLOSED NETWORK WITH NO CONTENDING ACKNOWLEDGEMENTS.

The lower bound is obtained as follows: Consider the case where only one message circulates (i.e. N=1), then the messge delay, T, is $3/\mu C$, neglecting nodal processing delays. As the number of messages increases, there will be some number such that at least one of the queues will become congested so that its output link is never empty, and hence, for this node,

$$\lambda = \mu C \quad .$$

Using Little's equation, the total average message delay is always given by

$$T = N/\lambda$$

where λ is the traffic for a single loop network.

When $\lambda = \mu C$, T is given by

$$T = N/\mu C$$

and this then becomes an asymptotic result.

The two asymptotes $3/\mu C$ and $N/\mu C$ intersect at N=3, and together the curves form a lower bound on average message delay. The lower bound and the result for exponentially distributed message lengths are plotted for message lengths of 500, 1000, 1500, and 2000

bits in Figures 3.5.2 through 3.5.5.

3.6 Experiment Four

This experiment uses the two node configuration of Figure 3.2.1a, differing from Experiment Two in that the network is closed. The new Experiment also differs from Experiment Three since acknowledgements mix and queue with data traffic. Like Experiment Three, the traffic generator assures that N messages are circulating in the network, where N is a variable parameter.

Figure 3.6.1 gives a queueing model for Experiment Four including the effect of acknowledgement traffic. The parameter μ' is computed in the same way as for Experiment Two and is given by

$$1/\mu' = \frac{1}{2}(1/\mu + 1/\mu_{ack})$$

An analysis of the queueing network of Figure 3.6.1 is given in Appendix 3.2, assuming exponentially distributed service times. The mean average message delay is determined to be given by

$$T = \left(\frac{N}{\mu C}\right)\left(\frac{N+1-(N+2)X + X^{N+2}}{NX-(N+1)X^2+X^{N+1}}\right)$$

where $X = \mu'/2\mu$. If N is much larger than 1, however, the expression for average message delay reduces to $T = N/\mu' C$,

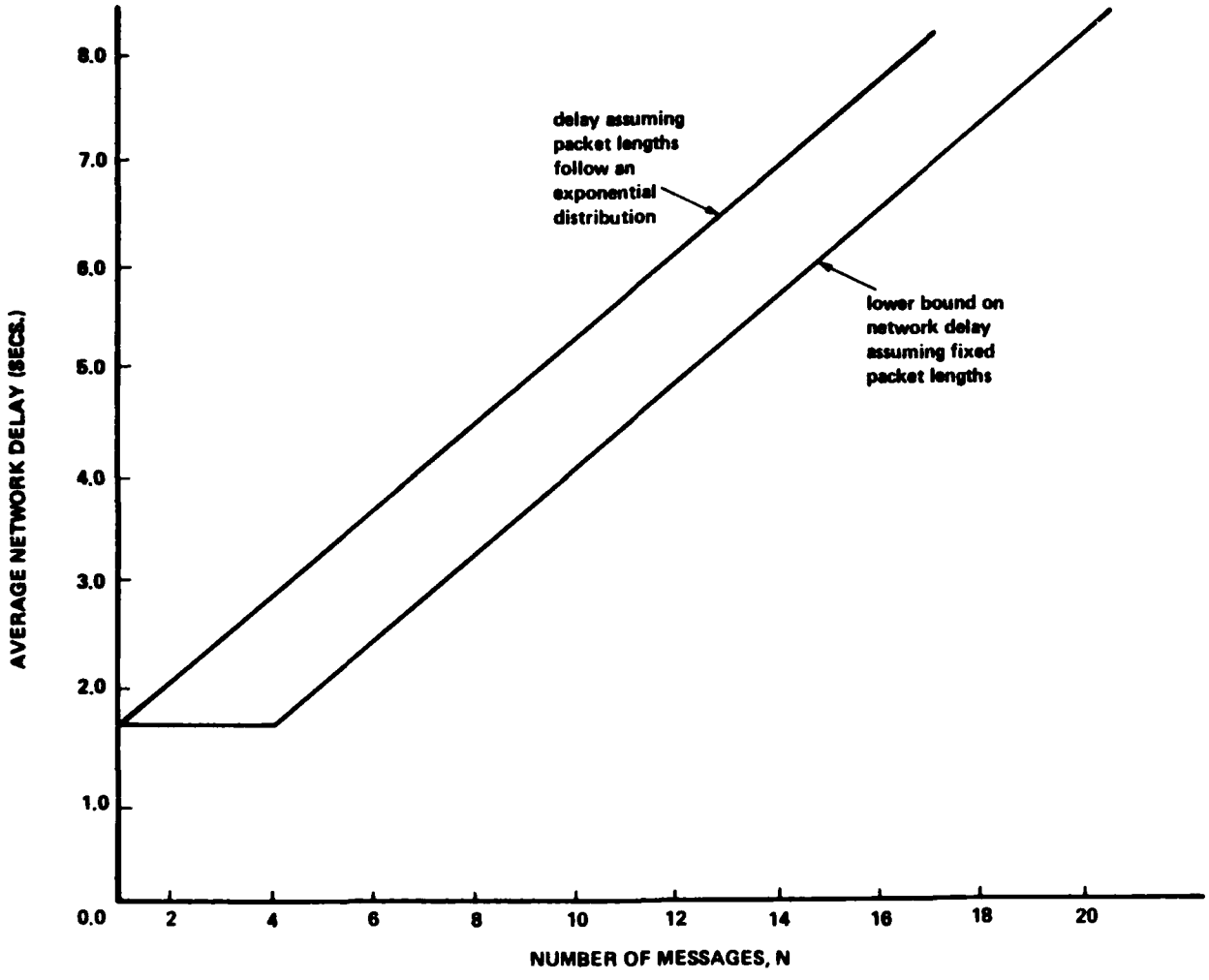


FIGURE 3.5.2. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 500 BITS, C = 1200 BITS/SEC., K = 0.

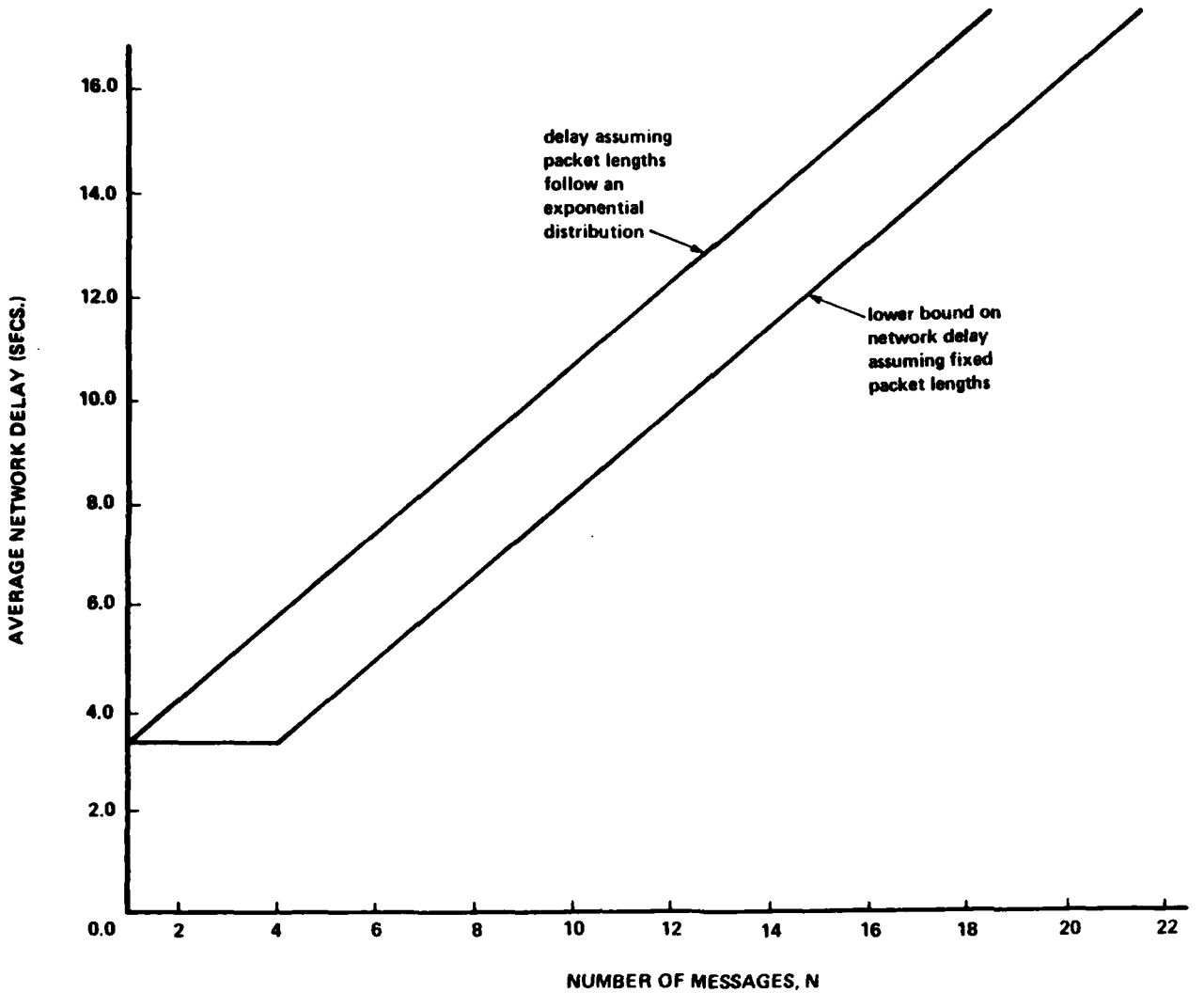


FIGURE 3.5.3. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 1000 BITS, C = 1200 BITS/SEC., K = 0.

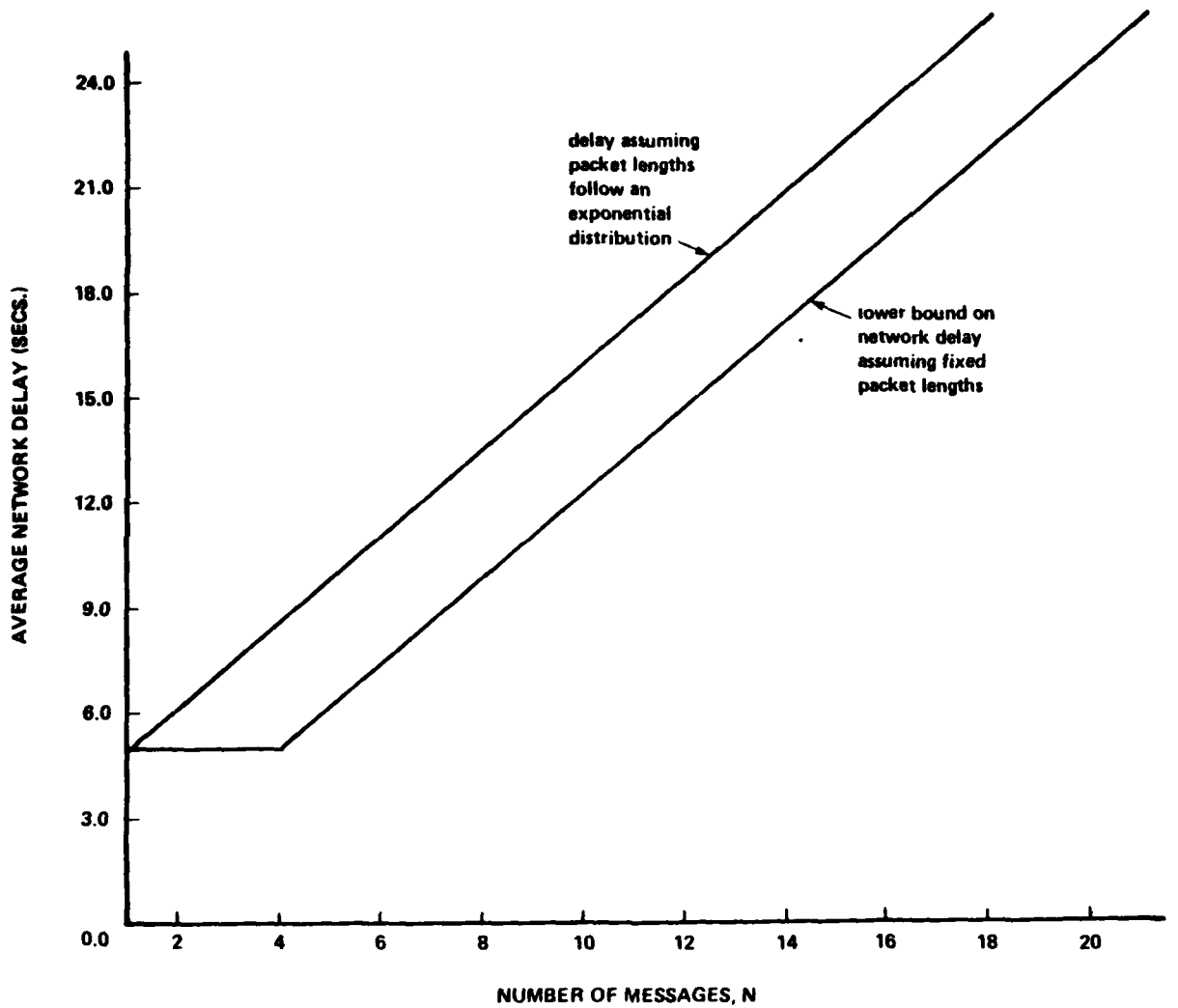


FIGURE 3.5.4. *AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK* FOR MESSAGE LENGTH= 1500 BITS, C = 1200 BITS/SEC., K = 0.

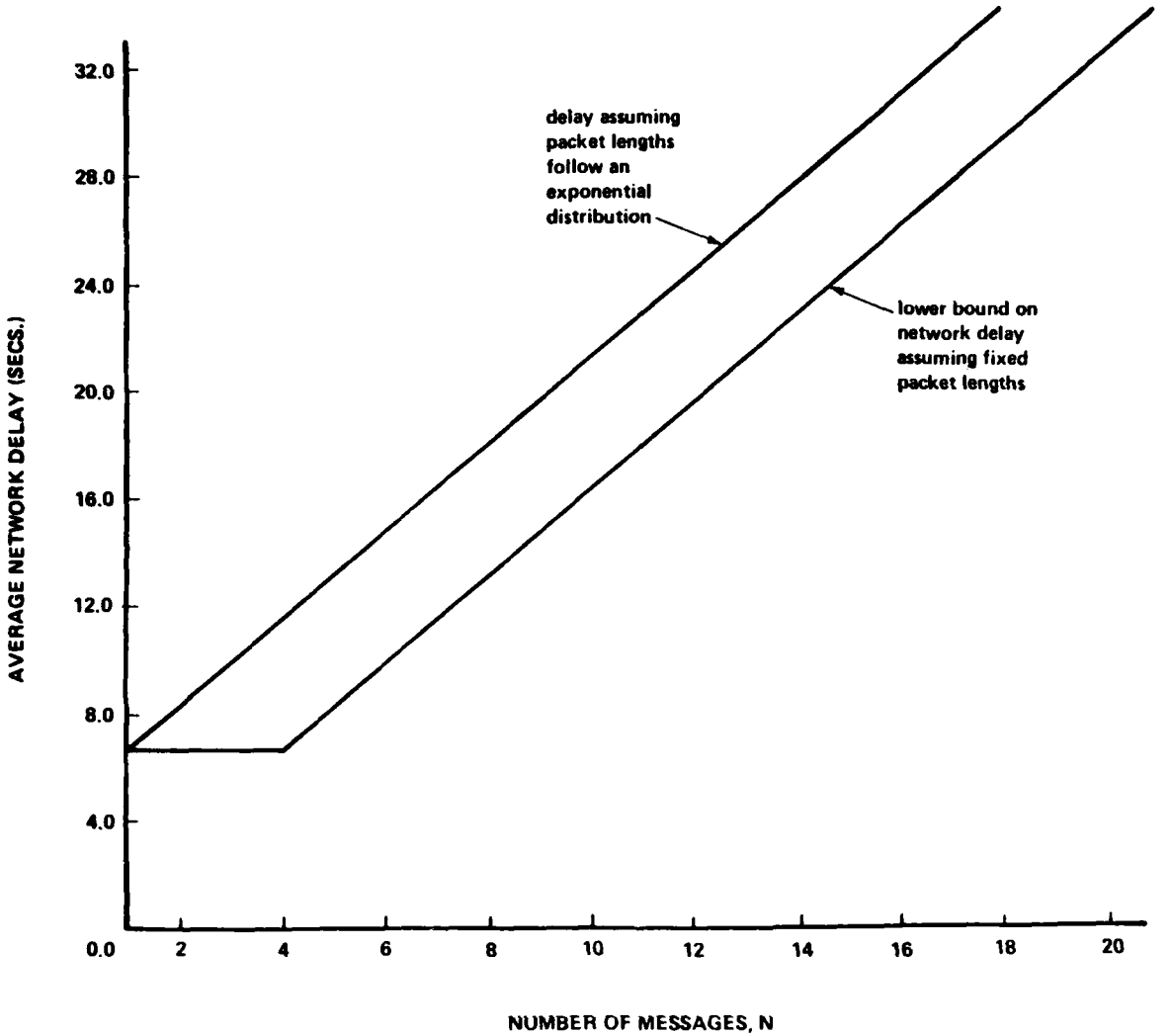


FIGURE 3.5.5. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 2000 BITS, C = 1200 BITS/SEC., K = 0.

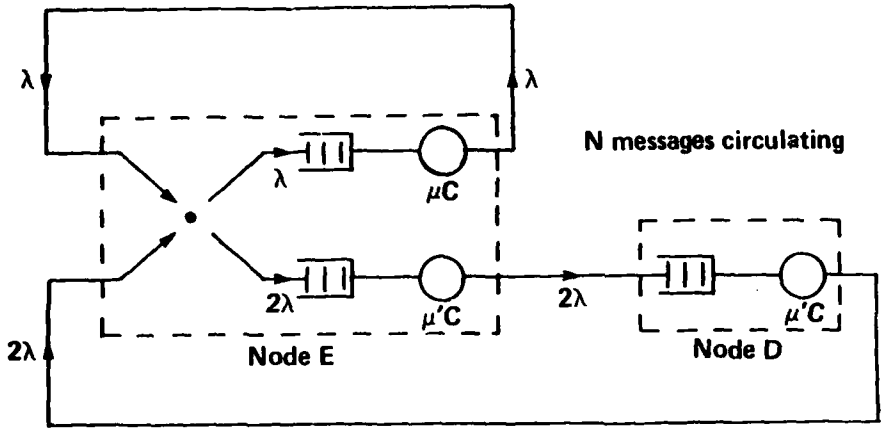


FIGURE 3.6.1. QUEUEING MODEL INCLUDING ACKNOWLEDGEMENT TRAFFIC.

and the curves plotted from the exact expression show that this expression, in fact, holds reasonably well for all values of N.

Lower bounds on the average message delay for the network of Figure 3.6.1, assuming constant message lengths, can be obtained in the same manner as for Experiment Three.

The case of N=1 is again considered first, and the resulting average message delay depends on the relative priorities assigned to acknowledgement and data traffic on the channel from Node B to Node A. The results are as follows:

data has priority: $T = 3/\mu C$

ack has priority:

$$T = 3/\mu C + 1/\mu_{ack} \quad C$$

neither has priority:

$$T = 3/\mu C + 1/2 \ \mu_{ack} \quad C \ .$$

At the extreme of large N, the queues which become congested first must be identified. Considering the expression for $1/\mu'$ shows that

$$1/\mu < 2/\mu'$$

and hence,

$$\lambda/\mu C \ < 2\lambda/\mu' C \ .$$

It follows that congestion occurs first at those queues for which average message length is $1/\mu'$. At these queues λ approaches $\mu' C/2$, for which value T is given by

$$T = 2N/\mu'C \quad .$$

Assuming that data has priority over acknowledgements, the piecewise linear bound for T becomes

$$T = \begin{cases} 3/\mu C & N \leq 3\mu'/2 \\ 2N/\mu'C & N > 3\mu'/2 \end{cases} .$$

Figures 3.6.2 through 3.6.5 each give three curves for average message delay versus N, namely: the curves obtained assuming exponential message length distributions, the piecewise linear bound developed above and a curve which ignores the effect of acknowledgements, (derived as in Experiment Three). The Figures use message lengths of 500, 1000, 1500 and 2000 bits.

3.7 Experiment Five

Experiment Five is designed to evaluate the finite storage model. To focus on the finite storage effect, it seems reasonable to

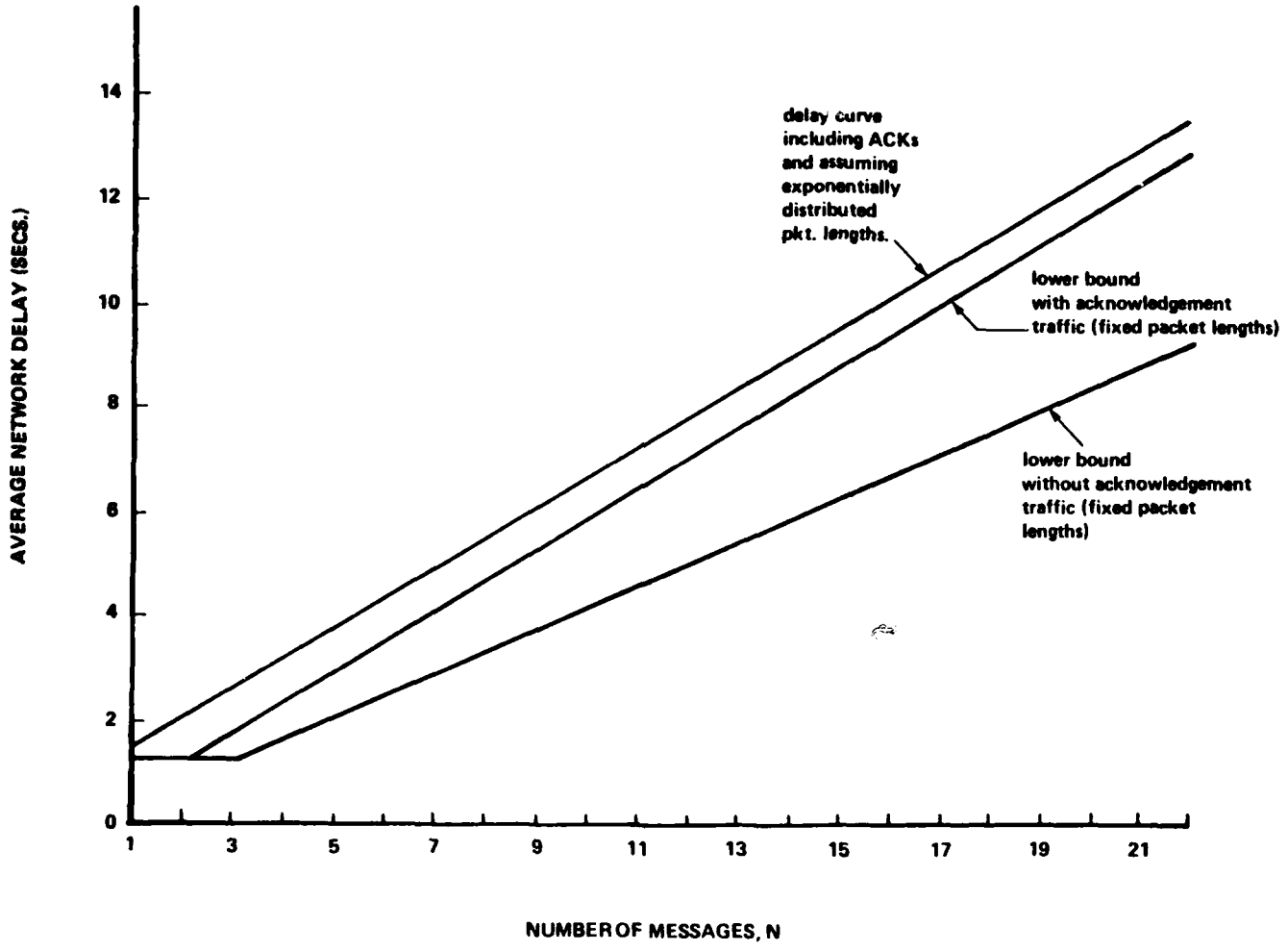


FIGURE 3.6.2. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 500 BITS, C = 1200 BITS/SEC., K = 0.

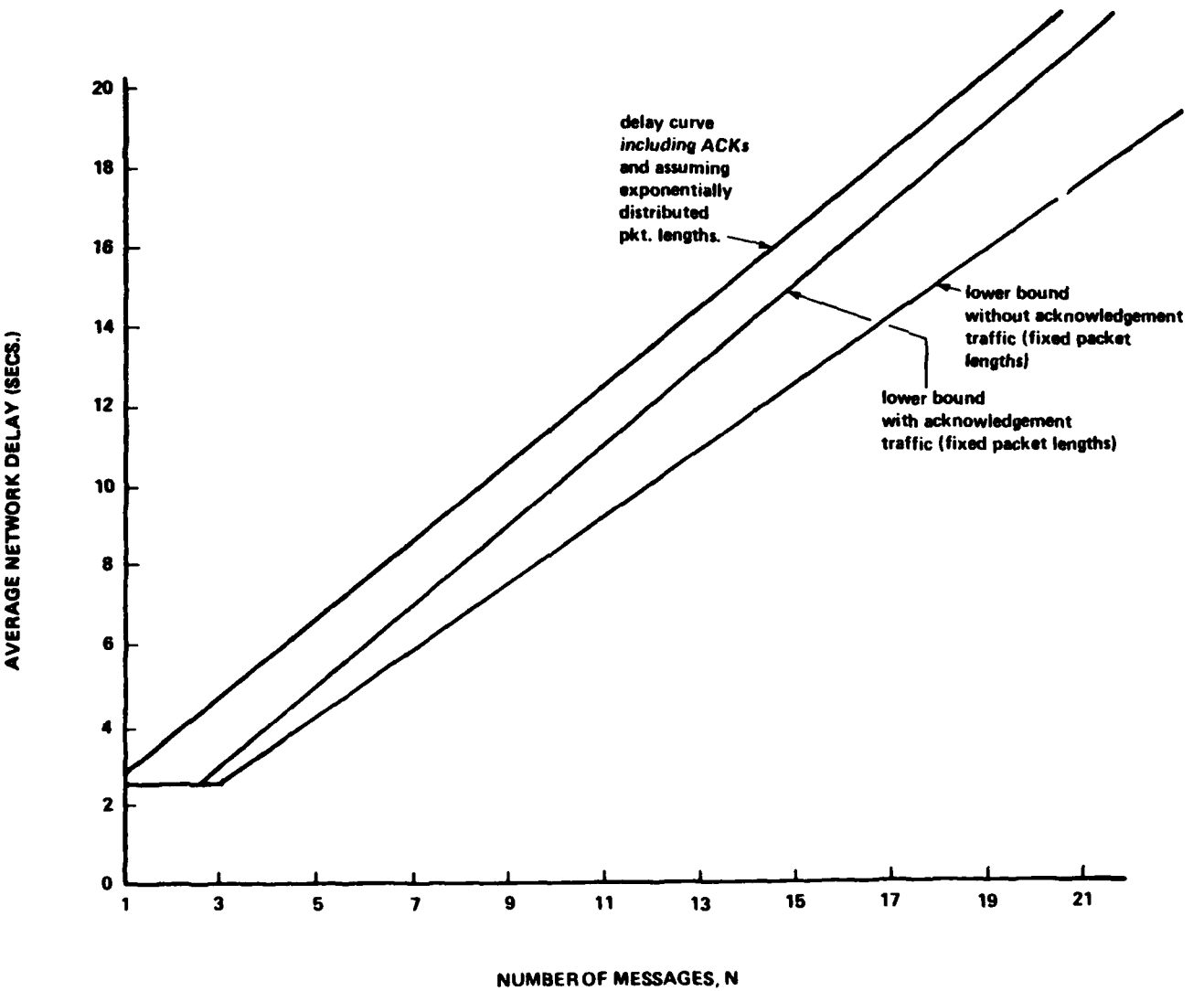


FIGURE 3.6.3. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 1000 BITS, C = 1200 BITS/SEC., K = 0.

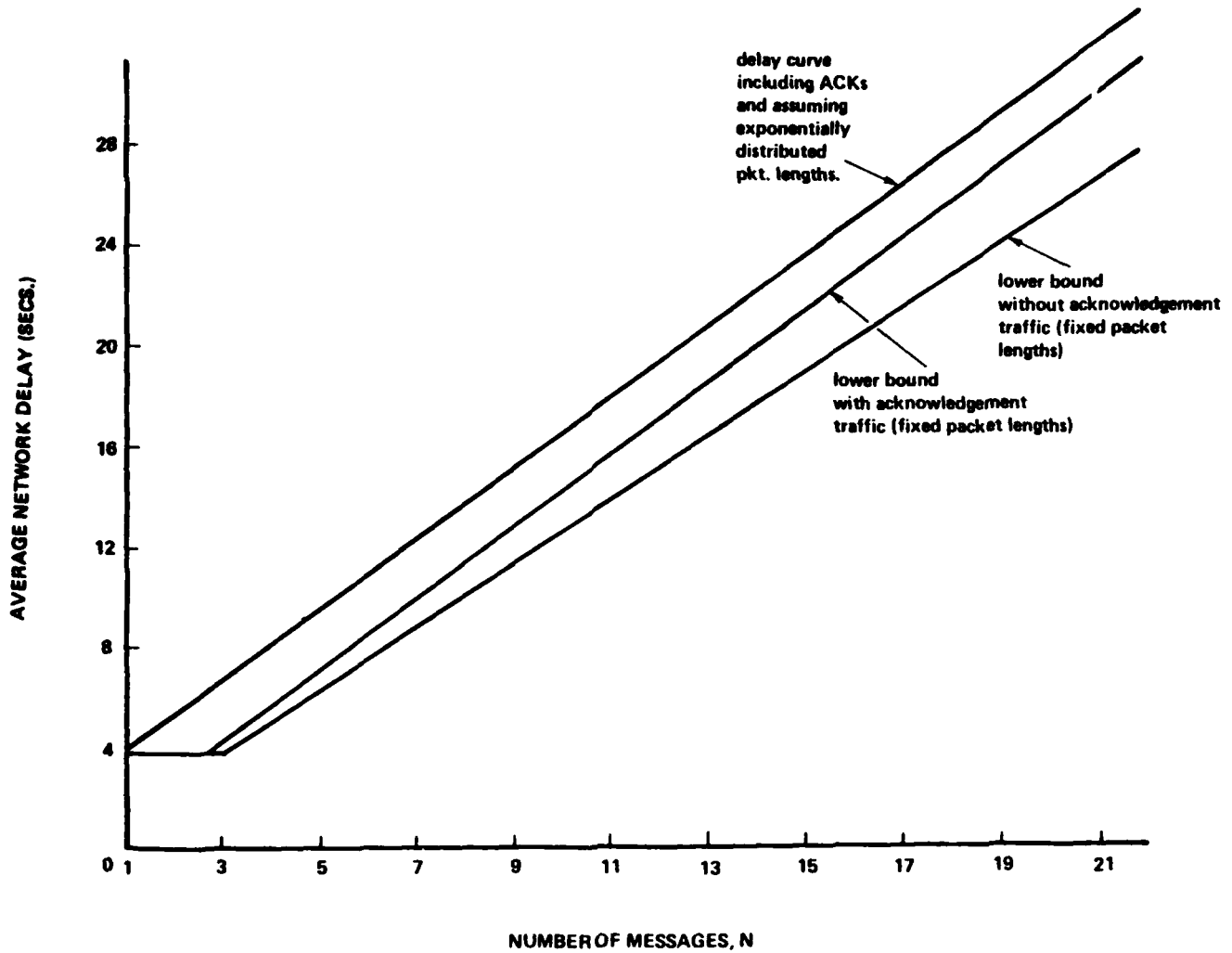


FIGURE 3.6.4. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 1500 BITS, C = 1200 BITS/SEC., K = 0.

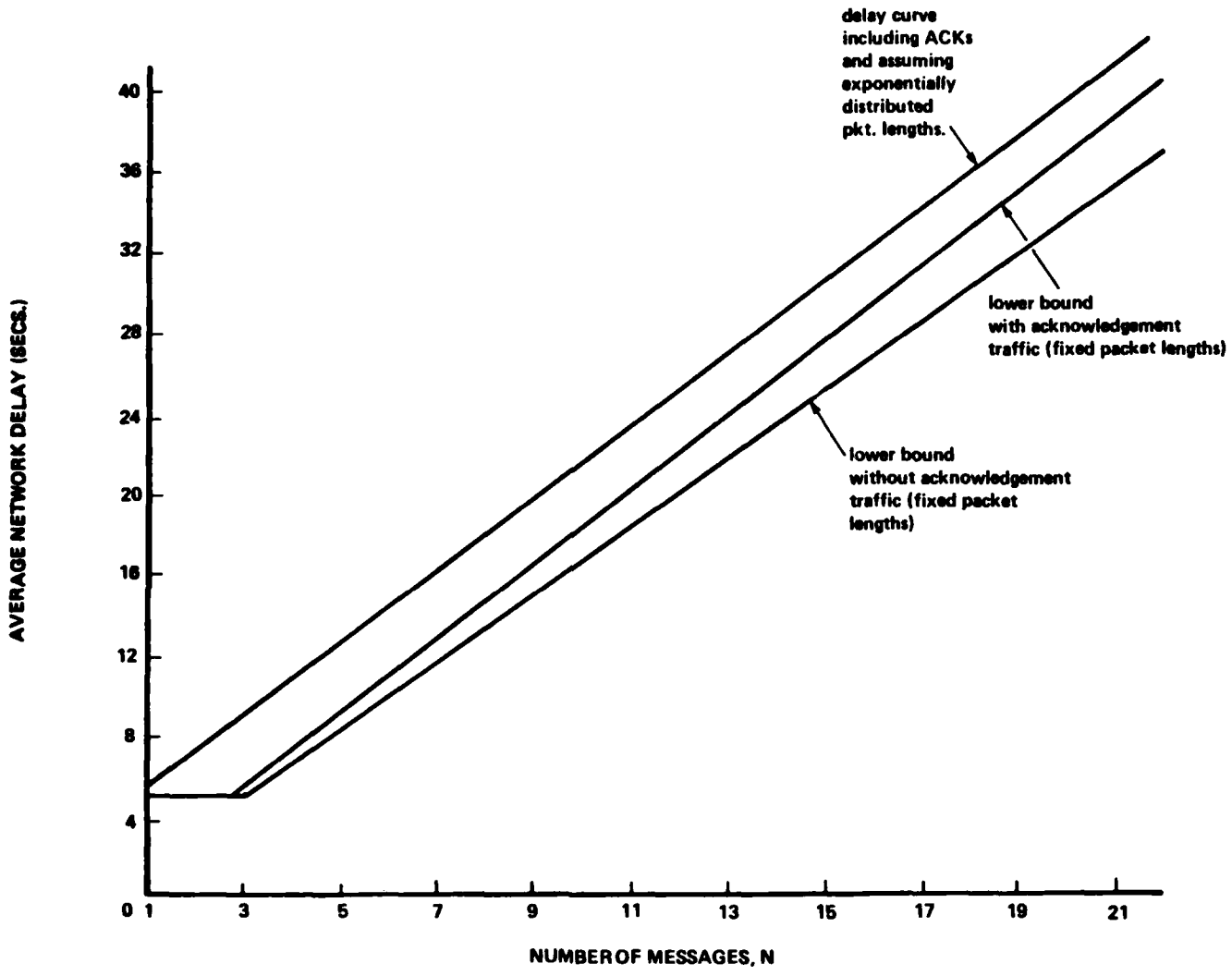


FIGURE 3.6.5. AVERAGE MESSAGE DELAY VERSUS NUMBER IN NETWORK FOR MESSAGE LENGTH = 2000 BITS, C = 1200 BITS/SEC., K = 0.

exclude acknowledgements. Thus the three node configuration of Figure 3.2.1 can be used if the buffers required by acknowledgements are ignored or either of the two or the three node configurations can be used with acknowledgements deleted from the physical system.

As noted in discussing the analytical approach, it is desirable to keep the number of network states small so as to make visualization possible in a three dimensional model and to illustrate principles without the requirement for extensive numerical calculation.

In the physical network, the requirement is to have an extremely small packet storage capacity at each node. This can be accomplished by having very long packets (so that each requires a large number of buffers) or by artificially reducing the number of available buffers at each node.

It is assumed that packets which arrive from the traffic generator have a Poisson distribution and are blocked from entering the network when all of the buffers at Node A are full. It is further assumed that packets in transit through the network are dropped if the node to which they are addressed has its buffers full. It is also assumed that no storage space is pre-allocated for any specific output link.

The analysis is carried out for a two node network with storage available for two packets at Node A and only one packet at Node B. A queueing model for the network is given in Figure 3.7.1. The

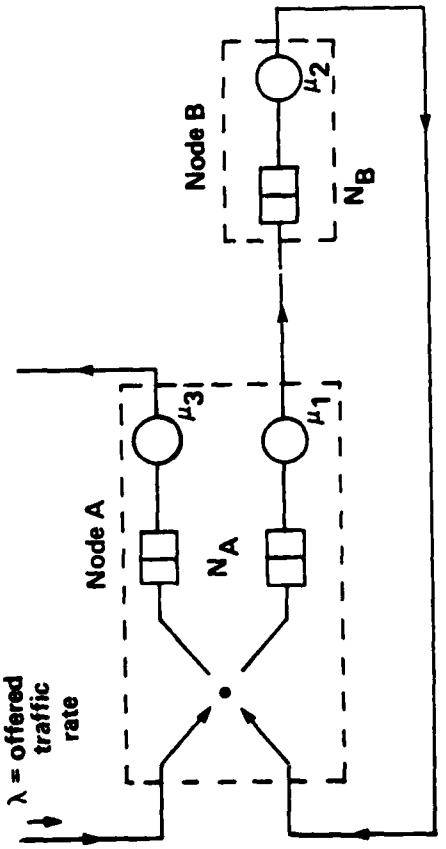


FIGURE 3.7.1. QUEUEING MODEL
N_A = STORAGE CAPACITY OF NODE A
N_B = STORAGE CAPACITY OF NODE B.

analysis could be extended relatively easily to storage for a few more packets; however, practical storage capacities would require extensive numerical computation.

With the stated assumptions and flow control policies, the finite Markov chain for the network shown in Figure 3.7.2 can be developed.*

The balance equations and the normalization equation can be solved for the state probabilities. Table 1 in Appendix 3.3 gives the numerical results for several values of $\rho = \lambda / \mu C$.

From the state probabilities, the blocking probabilities P_{B1}, P_{B2} and P_{B3} may be evaluated; these give the probability of an external packet being blocked or a transit message being dropped.

The throughput, γ, of the network may be evaluated to obtain:

$$\gamma = \lambda \prod_{i=1}^{3} (1-P_{Bi})$$

or

$$\gamma = \lambda (1-P_{B1})(1-P_{B2})(1-P_{B3}) \quad .$$

---

*Note that more assumed storage and hence, more states would result in an excessively complicated diagram for the Markov chain.

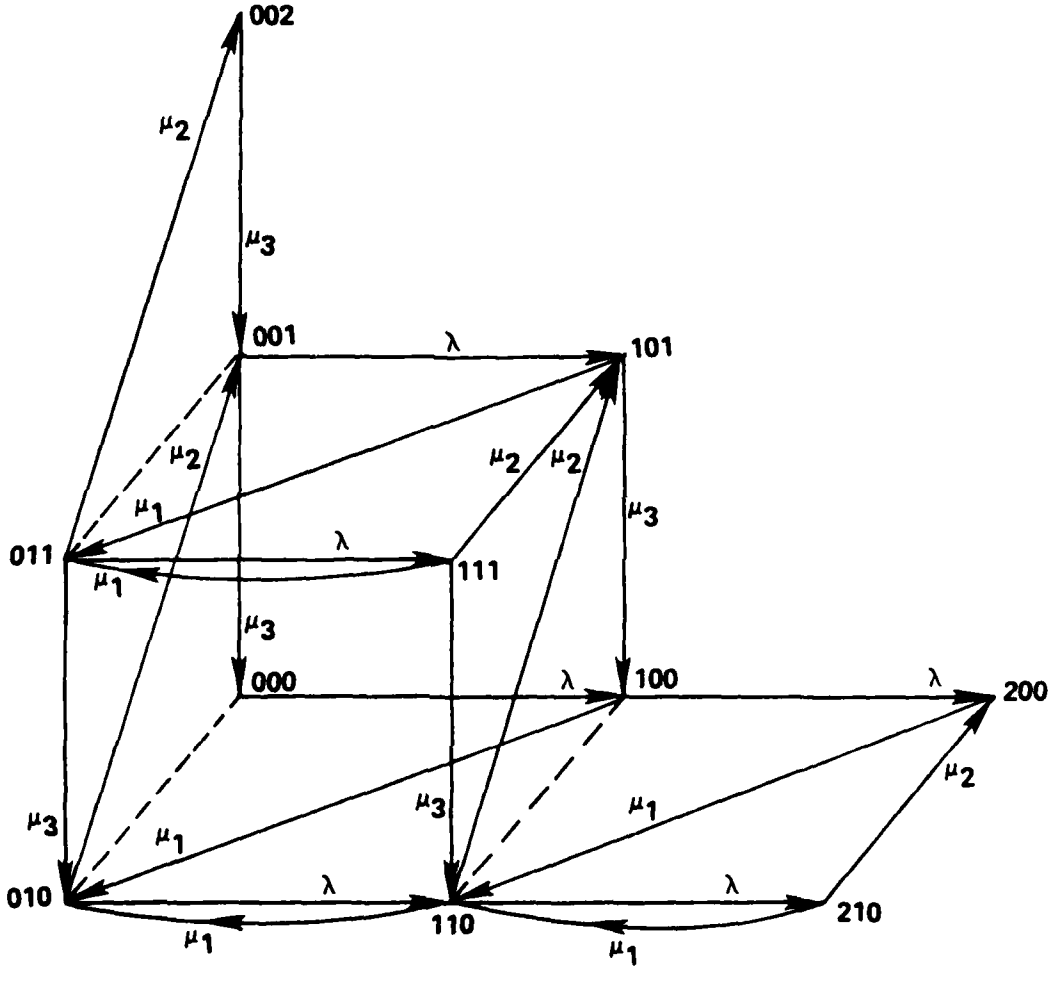


FIGURE 3.7.2. MARKOV CHAIN FOR FINITE STORAGE MODEL OF EXPT. 5 WITH $N_A = 2$, $N_B = 1$.

For this network, $P_{B1} = P_{B3}$.

Figure 3.7.3 gives the throughput versus offered load for this network under the stated assumptions and flow control policies for $\mu C = 10$, an arbitrary value. Figure 3.7.4 shows how the average number of packets in the network varies with ρ, the utilization factor.

A network such as that considered does not give a product-form solution, i.e., local balance equations cannot be written for the Markov chain. Hence, no known closed form expression can be written for the state probabilities. Since the Liu Reduction method is primarily advantageous in cases for which the product-form solution applies, it is not used here.

In reviewing the analysis of Appendix 3.3, note that to obtain the throughput, only the blocking probabilities need to be evaluated. For this particular network, eight of the eleven state probabilities are involved in the blocking probabilities. However, as the storage sizes N_A and N_B increase, the relative number of state probabilities required decreases. For cases when the product-form solution does not apply, it is generally computationally easier to calculate the state probabilities by matrix inversion, or by a sparse matrix technique, than to use the Liu Reduction Method.

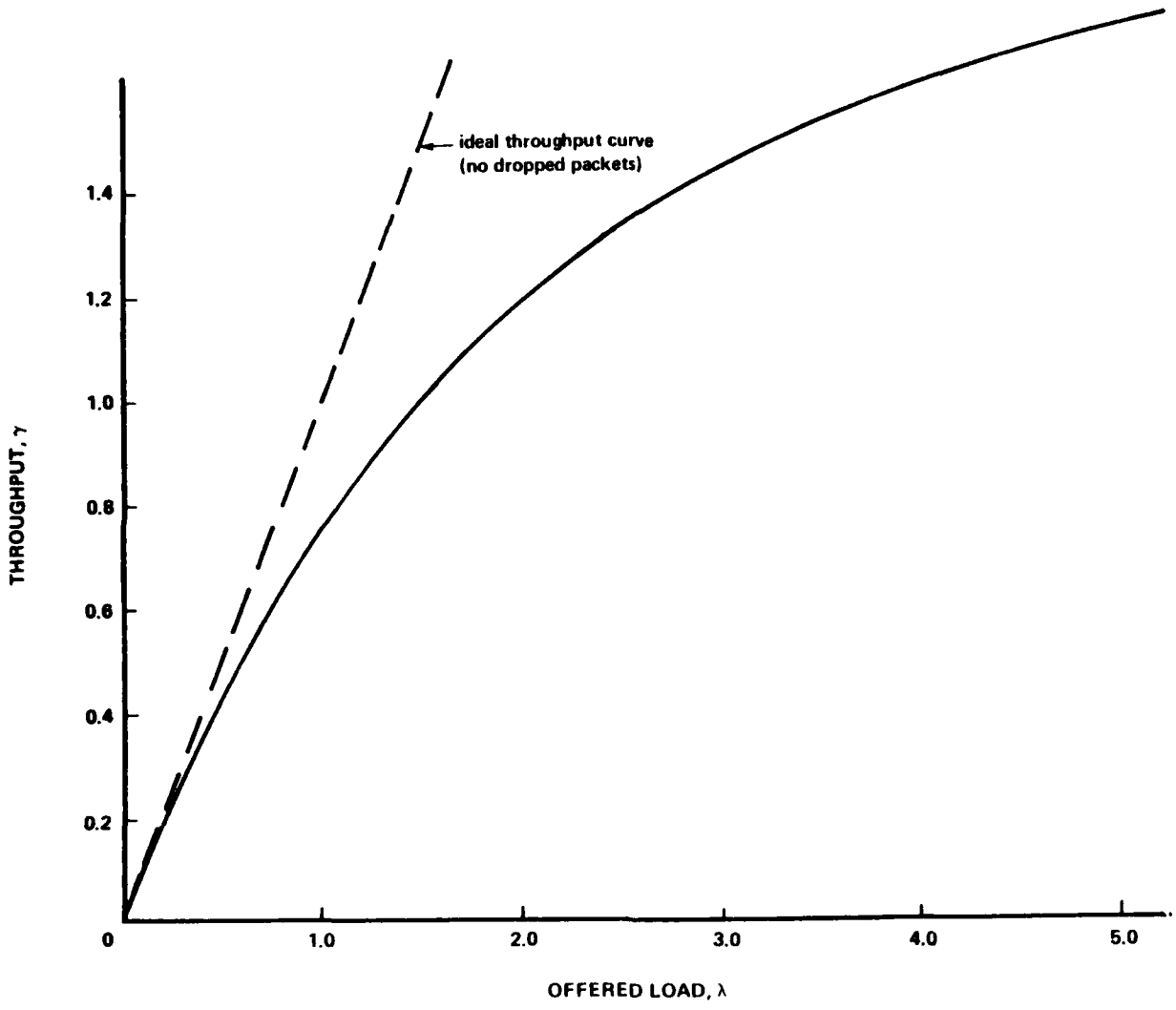


FIGURE 3.7.3. THROUGHPUT VERSUS OFFERED LOAD CHARACTERISTIC ($\mu C = 10$).

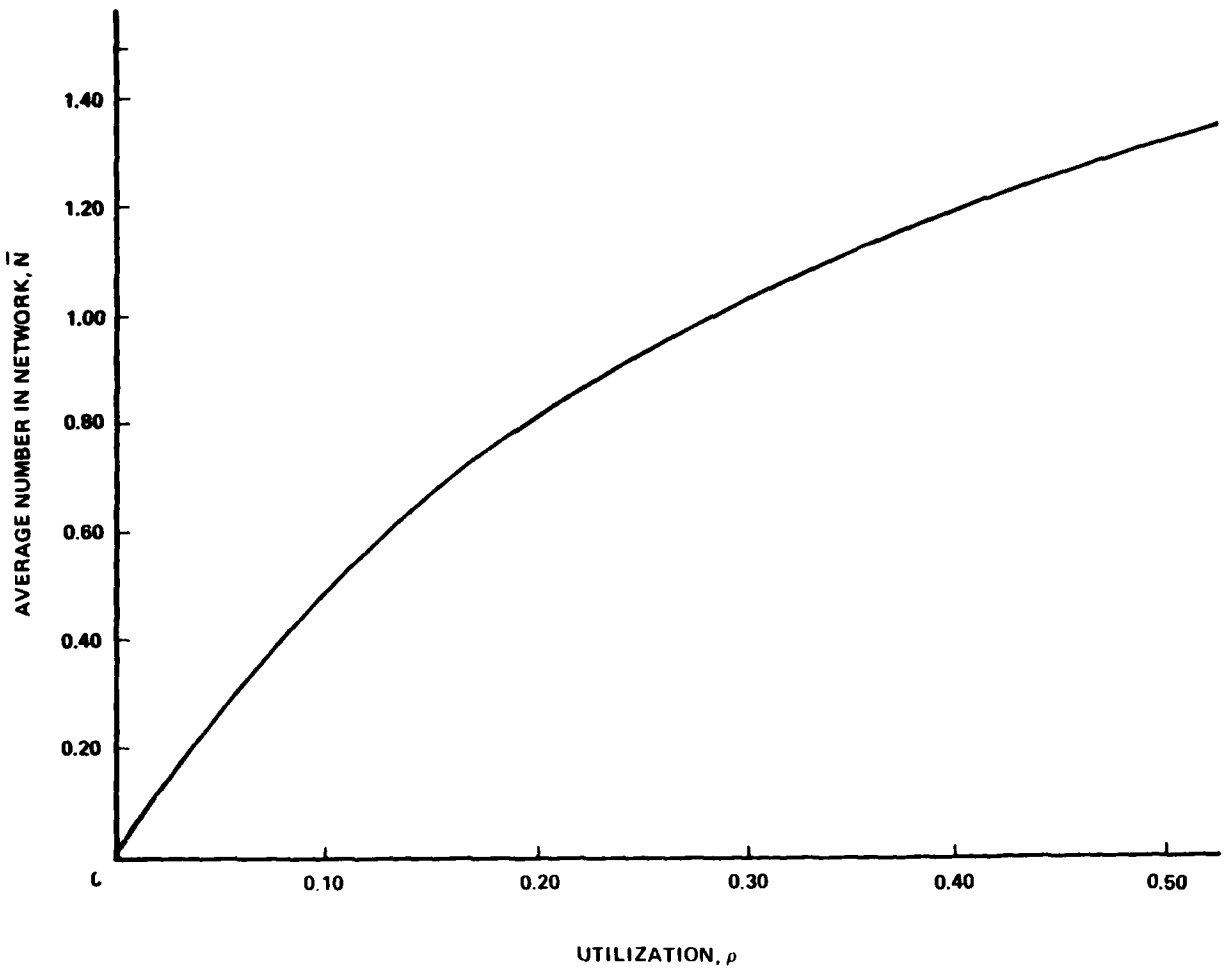
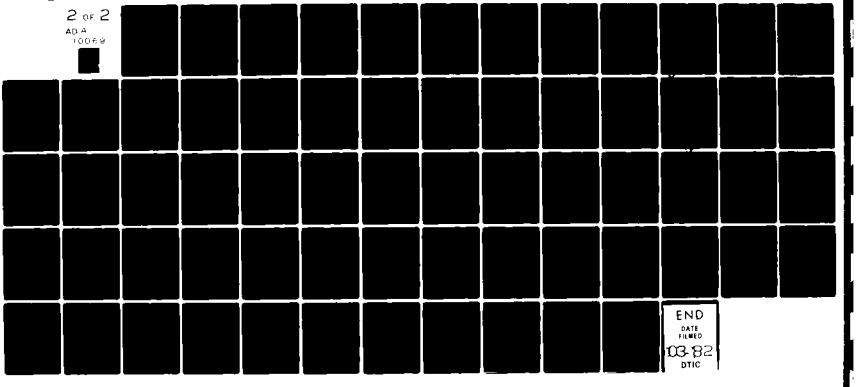


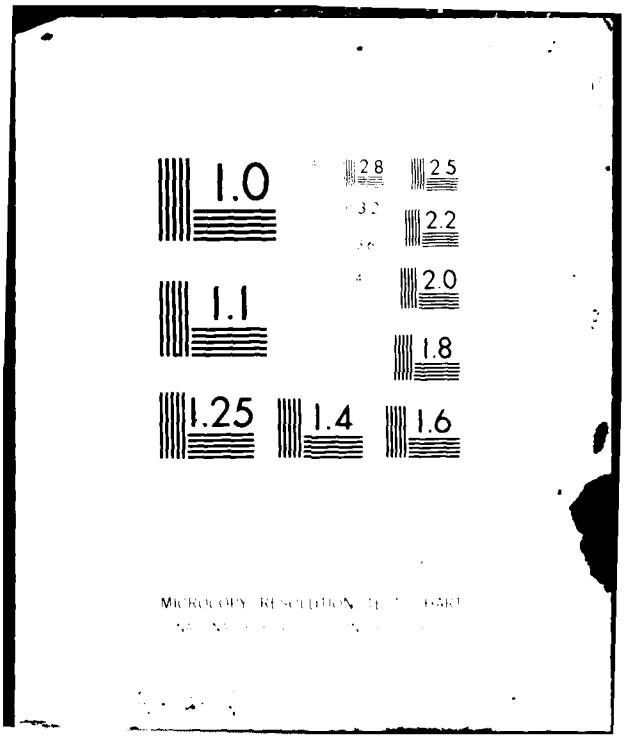
FIGURE 3.7.4. NUMBER IN NETWORK VERSUS UTILIZATION CHARACTERISTIC.

AD-A110 069 GEORGIA INST OF TECH ATLANTA SCHOOL OF ELECTRICAL EN--ETC F/G 9/2
STUDY OF A HETEROGENEOUS DISTRIBUTED MICROCOMPUTER NETWORK USIN--ETC(U)
JUL 81 J L HAMMOND, J H SCHLAG, D K LAM DAA629-80-K-0009
UNCLASSIFIED E21-616 NL

2 OF 2
AD A
110069



END
DATE FILMED
10-82
DTIC


1.0
2.8
2.5
3.2
2.2
2.0
1.1
1.8
1.25
1.4
1.6

APPENDIX 3.1: ANALYSIS OF CLOSED TANDEM NETWORK WITH IDENTICAL SERVERS

Consider the situation where N packets are circulating among K queues which are connected in tandem. The service times of each server are exponentially distributed with mean $1/\mu C$.

It can be easily shown that the total number of states in the ensuing Markov chain is

$$\binom{N+K-1}{K-1} .$$

Taking any arbitrary queue to be the first queue, the number of states with this queue empty is

$$\binom{N+K-2}{K-2} .$$

Such closed queueing networks give a "product-form" solution, and since each queue has identical packet arrival and service rates,

$$P_{k_1,\ldots k_k} = G(N) \left(\frac{\lambda}{\mu C}\right)^{k_1} \ldots . \left(\frac{\lambda}{\mu C}\right)^{k_k}$$

with $k_1 + \ldots + k_k = N$, $G(N)$ a normalization constant and λ the unknown arrival rate.

Clearly, all states are equally likely. Hence the probability of queue 1 being empty is

$$\binom{N+K-2}{K-2} \Big/ \binom{N+K-1}{K-1} = \frac{K-1}{N+K-1} \quad .$$

Hence the probibility of queue 1 being empty is

$$1 - \frac{K-1}{N+K-1} = N/(N+K-1) \quad .$$

The arrival rate to each queue, equivalent to network throughput, is then given by

$$\lambda = \mu CN/(N+K-1) \quad .$$

Using Little's Law, the total average message delay, T, in traversing all queues is given by

$$T = N/\lambda = (N+K-1)/\mu C \quad .$$

APPENDIX 3.2: ANALYSIS OF THE CLOSED NETWORK OF FIGURE 3.6.1

Consider such a general network with N packets circulating among K identical queues (K-1 nodes). Due to the presence of acknowledgements, all queues except the final "output" queue have a service rate $\mu' C$ and input traffic rate 2λ. The output queue has input traffic rate λ and service rate μC.

As for the network of Figure 3.5.1, there are

$$\binom{N+K-1}{K-1} \quad \text{states of which} \quad \binom{N+K-2}{K-2}$$

Have the output queue empty. In this case all states are not equally likely.

The probability, P_{KO}, of the output queues being empty can be computed in a straightforward way to obtain

$$P_{KO} = \frac{\binom{N+K-2}{K-2}}{\sum_{i=0}^{N} \binom{N+K-2-i}{K-2} \left(\frac{\mu'}{2\mu}\right)^i} .$$

In the network under consideration there are three queues, so that K = 3 and

$$P_{K0} = \frac{1}{\sum_{i=0}^{N+1} (N+1-i)(\mu'/2\mu)^i} = \frac{(1-X)^2(N+1)}{N+1-NX-2X+X^{N+2}}$$

where $X = \mu' / 2\mu$.

The throughput of this network, λ , is given by

$$\lambda = \mu C \ (1-P_{K0})$$

By Little's Law, the average network delay, T, is given by:

$$\begin{aligned} T &= \frac{N}{\lambda} \\ &= \frac{N}{\mu C(1-P_{K0})} \\ &= \frac{N}{\mu C} \cdot \frac{N+1 - NX - 2X + X^{N+2}}{NX - (N+1)X^2 + X^{N+2}} \end{aligned}$$

Since X < 1 always, the term X^{N+2} may be neglected except for small values of N. Hence,

$$\begin{aligned} T &= \frac{N}{\mu CX} \left\{1 + \frac{1-X}{N-(N+1)X}\right\} \\ &= \frac{2N}{\mu' C} \quad \text{for large enough N .} \end{aligned}$$

APPENDIX 3.3: BALANCE EQUATIONS FOR THE MARKOV CHAIN OF FIGURE 3.7.2

The balance equations are:

$$\lambda \rho_{001} + \mu_2 \rho_{110} + \mu_2 \rho_{111} = (\mu_1+\mu_3) \rho_{101}$$

$$\mu_2 \rho_{010} + \mu_3 \rho_{002} = (\lambda+\mu_3) \rho_{001}$$

$$\mu_3 \rho_{001} = \lambda \rho_{000}$$

$$\lambda \rho_{000} + \mu_3 \rho_{101} = (\lambda+\mu_1) \rho_{100}$$

$$\mu_1 (\rho_{100}+\rho_{110}) + \mu_3 \rho_{011} = (\lambda+\mu_2) \rho_{010}$$

$$\mu_3 \rho_{111} + \lambda \rho_{010} + \mu_1 \rho_{210} + \mu_1 \rho_{200} = (\mu_1+\mu_2+\lambda) \rho_{110}$$

$$\lambda \rho_{110} = (\mu_1+\mu_2) \rho_{210}$$

$$\lambda \rho_{100} + \mu_2 \rho_{210} = \mu_1 \rho_{200}$$

$$\mu_1 \rho_{101} + \mu_1 \rho_{111} = (\lambda+\mu_2+\mu_3) \rho_{011}$$

$$\mu_2 \rho_{011} = \mu_3 \rho_{002}$$

$$\lambda \rho_{011} = (\mu_1+\mu_2+\mu_3) \rho_{111} \ ,$$

where

$$\sum_{\text{all } i,j,k} \rho_{ijk} = 1 \ ,$$

the ρ_{ijk} are the stationary probabilities of having k messages in queue 1, etc., and μ_i in these equations denotes the $\mu_i C$ from the text.

The solutions of these equations for specified values of $\rho = \lambda/\mu$ are given in Table 1. Additional quantities required are:

$$P_{B1} = \text{probability of blocking at queue 1}$$

$$= \rho_{101} + \rho_{210} + \rho_{200} + \rho_{002} + \rho_{111}$$

$$= P_{B3}$$

$$P_{B2} = \rho_{210} + \rho_{111} + \rho_{011} + \rho_{110} + \rho_{010}$$

$$\bar{N} = \text{average number of packets in the network}$$

$$= \sum_{\text{all } i,j,k} (i+j+k)\, \rho_{ijk}$$

$$= \rho_{001} + \rho_{100} + \rho_{010} + 2\,(\rho_{101} + \rho_{110} + \rho_{200} + \rho_{002} + \rho_{011}) + 3\,(\rho_{111} + \rho_{210})$$

Note that with the restrictions $N_A = 2$, $N_B = 1$, there are 12 possible states from this network. However, only eleven of these are reachable; state (012) cannot be reached for any of the others, and hence $\rho_{012} = 0$.

TABLE 1. State probabilities for different $\rho = \lambda/\mu$.

λ/μ	0.05	0.10	0.15	0.20	0.25	0.30
ρ_{000}	0.789994	0.639627	0.527388	0.441902	0.375144	0.322037
ρ_{210}	0.001061	0.003668	0.007234	0.011398	0.015927	0.020664
ρ_{111}	0.000009	0.000063	0.000190	0.000403	0.000708	0.001109
ρ_{010}	0.040933	0.068422	0.087176	0.100017	0.108733	0.114503
ρ_{100}	0.058772	0.094401	0.115968	0.128675	0.135658	0.138876
ρ_{200}	0.004000	0.013108	0.024629	0.037133	0.049842	0.062327
ρ_{001}	0.039500	0.063933	0.079108	0.088381	0.093786	0.096611
ρ_{110}	0.042437	0.073361	0.096454	0.113981	0.127417	0.137762
ρ_{011}	0.000542	0.001903	0.003798	0.006039	0.008499	0.011092
ρ_{002}	0.000542	0.001903	0.003798	0.006039	0.008499	0.011092
ρ_{101}	0.022211	0.039909	0.054255	0.066030	0.075786	0.083927
P_{B1}	0.027823	0.058651	0.090110	0.121003	0.150762	0.179119
P_{B2}	0.084982	0.147417	0.194852	0.231838	0.261284	0.285130
γ/λ	0.864809	0.755506	0.666582	0.593509	0.532766	0.481712
$\bar{N}$	0.281879	0.495317	0.670392	0.810920	0.928168	1.027709
$\frac{\bar{N}\lambda}{\gamma}$	0.325944	0.659580	1.005716	1.366315	1.742168	2.133451

4. EXPERIMENTAL RESULTS

4.1 Introduction

In this section the experimental results of the proposed experiments described in Section 3 are presented. The network was connected in a numb - of configurations and the performance monitoring techniques described in Section 5 were used to collect information about message delays in the system. The Nova CS-20 system was used to generate traffic to the network in different statistical distributions. Because of hardware limitations of the CS-20 serial i/o interface the rate at which the network could be made to accept data was very limited and experimental results was limited to the lower part of the curves in Section 3.

4.2 Explanation of Hardware Limitations

The DTR (data terminal ready)

And DSR (data set ready) interface lines on the CS-20 computer are not connected to the data section of the serial interface. This forced the CS-20 CPU to monitor and control these lines and send the proper control signals to the data section. These lines are not available as an interrupt so that the control and monitor process must be maintained at a rapid interval to assure proper handshake operation. If the DTR from the network was lowered and returned

(message recieved by network) at a time when the CS-20 was not available to monitor the operation then the CS-20 would assume that the network was not ready to accept more data and would stop sending. The network would assume that the CS-20 had recieved the handshake protocol and would wait for the next character. This would stop all operation of the traffic generator and the traffic program would need to be cleared and restarted. Several software techniques where developed to detect when the system had missed the hardware protocol and automatically clear the hardware and continue operation. This software solution kept the system from terminating operation but required significant amount of time and resulted in the loss of one message.

The final system available to send traffic to the network was therefore limited in speed and would tend to lose messages as the speed limit was approached. This problem would tend to emphasis the importance of handshake protocol lines in the network serial interfaces.

4.3 Low Speed Network Data

The system was connected in different configurations and loaded with message traffic of different bit lengths. The system was configured with the traffic generator with one to three nodes. These configurations were run with and without local acknowledgments. Message lengths varied from 10 to 60 characters (100 to 600 bits).

The message data was collected as a table of departure times and arrival times of each message. Figure 4.1 shows a typical data table where the first column represents the number of the message being transmitted or recieved, the second column represents the number of bytes in a departing message, the third column represents the time of departure in counts, and the fourth column represents the time of arrival of the return message in counts. One count in the departure and arrival column represents a time interval of 1.8432 millisecond. Time delay can be determined by calculating the diffenence between departure and arrival. For example the time delay for message one equal 1870 - 859 = 1011 counts or 1.86 seconds.

The data obtained in the low speed experiments compared well with theoretical estimate as shown in the comparison values in Figure 4.2. This data was obtained by sending messages varing in length from 10 to 210 at 300 baud through the system shown in Figure 4.3.

TIMING DATA TABLE

RATE = UNIFORM - 300 BAUD

LENGTH = UNIFORM - 30 CHARACTERS/MESSAGE

1	30	950	0
2	30	1807	0
3	30	1141	0
4	30	3452	0
1	0	0	1570
2	0	0	2821
3	0	0	3356
4	0	0	4391

FIGURE 4.1 DATA TABLE

NO. OF BYTES PER MESSAGE	MEASURED DELAY	THEORETICAL DELAY
10	2.7 SEC.	2.7 SEC.
20	4.3 SEC.	4.5 SEC.
30	5.7 SEC.	5.8 SEC.
40	8.4 SEC.	8.5 SEC.
60	9.8 SEC.	9.8 SEC.
80	12.7 SEC.	12.5 SEC.
110	16.9 SEC.	16.9 SEC.
160	24.0 SEC.	24.0 SEC.
200	31.2 SEC.	31.0 SEC.

FIGURE 4.2 EXPERIMENTAL DATA EVALUATION

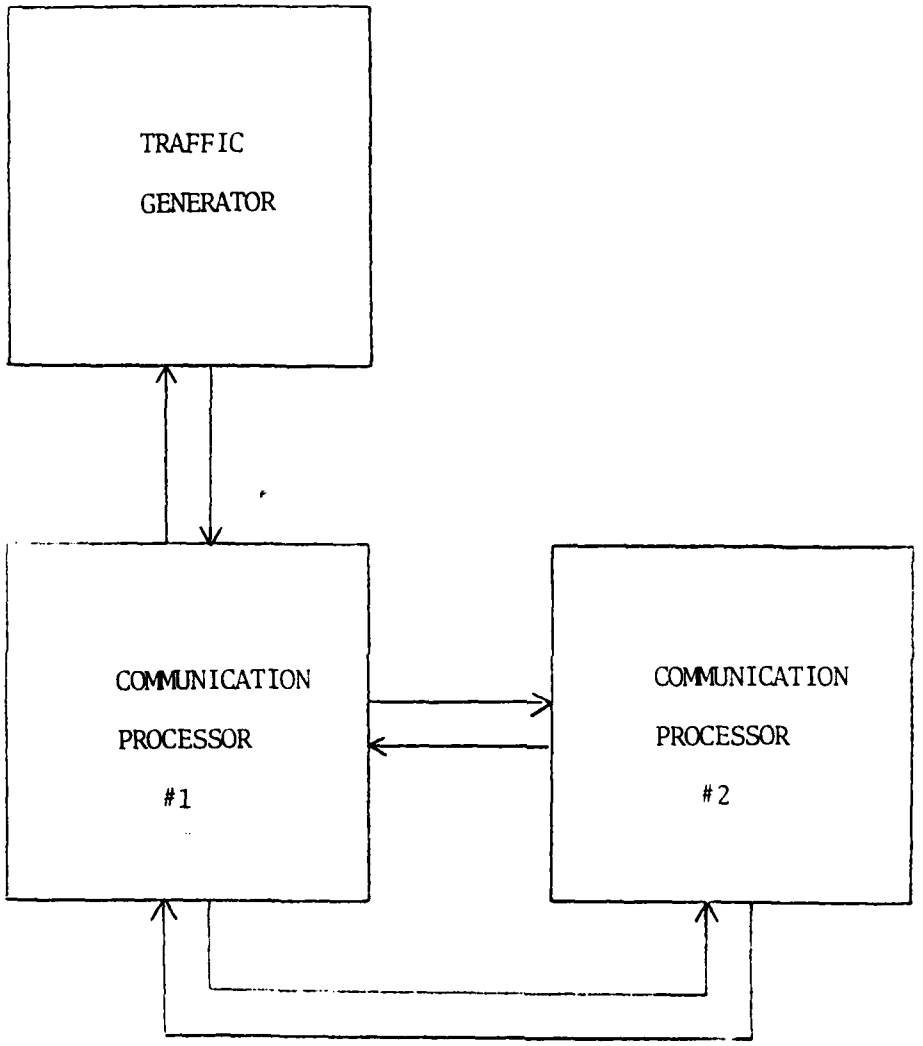

TRAFFIC
GENERATOR
COMMUNICATION
PROCESSOR
#1
COMMUNICATION
PROCESSOR
#2

FIGURE 4.3

5. Performance Monitoring

5.1 Introduction

In order to adequately evaluate the validity of the computer models presented in the previous sections, it was necessary to collect data regarding the actual performance of these networks and to compare that data with the performance as predicted by the models. Certain types of data regarding the network performance could be monitored externally by monitoring the traffic going into the network and the traffic coming back out of the network. For example, total network loop delay could be found by generating a message which is routed back to the original center for which total time between the entrance of the message into the network and its return is measured. Some types of performance data could not be measured externally and could be obtained only by monitoring events inside the communication nodes. An example of this type of data is the number of messages queued at any one particular node.

There are several techniques that can be used to retrieve internal performance data from the microprocessor communication nodes. One approach is to add portions to the communication node software so that the communication node continually sends out information regarding its status. This software monitoring approach has the advantage of requiring little or no additional hardware to achieve performance monitoring, but has the disadvantage of requiring a certain amount of software time which distracts from the

performance of the network, thereby affecting the performance of the system.

An approach that tends to minimize the interaction of the communication process and the monitoring process is the use of a combination of hardware and software. With this technique, small software routines with fast execution times are placed in the major software loops to pick up key information from the communication process without slowing down the process significantly. Hardware added to the system picks up this key information and transmits it to a central collection system. This is a compromise system, where hardware is added to minimize the effect of the software execution time, but does not totally eliminate the impact of the performance monitoring task on the communication task.

A third method uses hardware added to communication processors to measure the node performance without any intervention in the communication software. This technique, although more expensive, has the advantage of being totally non-invasive and therefore will not affect the system's performance in its attempt the measure it. In this research project, completely non-invasive hardware type monitoring was used to measure network performance to guarantee that the measuring process would in no way affect the performance of the network itself.

5.2 Monitoring Approach

The communication processors consist of the CPU, the ROM memory, containing the processor control software, the RAM memory, containing the message data with its associated queues and stack pointers and four serial I/O ports for transmitting and receiving data.

If care is taken in designing the software for the communication processing, selected memory locations within the random access memory will contain sufficient information to accurately reflect the performance of the communication node at any particular time. These locations include the current values of stack pointers, queue links, source and destination codes, etc.

The basic approach for implementing a hardware monitor for the communication network is to use a two port RAM memory with one of the two ports connected to the communication processor CPU for the normal storage and retrival of message data and its associated buffer information. The second port of the memory would allows an additional central processing unit to access the same memory space and retrieve information necessary for performance monitoring without interrupting the process of the communication CPU. The speed of the two port memory is sufficiently fast so that the central processors can access the memory with no degradation in speed to the communication process. In order to maintain the speed and versatility required by the communication process, the interconnection between the communication central processor and the port on the RAM memory is constrained to be the standard memory bus interface used by the communication central processor.

The central processor used for the performance monitoring of the monitor CPU does not require as rigid a time access to the memory and could be forced to wait several memory cycles for access to the RAM memory. Because of this flexibility, two techniques are employed and implemented from the monitor central processor to the other port of the RAM memory. The first technique is a time-shared data bus where both central processors are on the same memory bus and take turns accessing the memory. This technique requires time coordination between the two central processors and a memory that is sufficiently fast to be accessed by both central processors without loss of time to either processor. The second technique uses an I/O type interface between the monitor CPU and the memory. The monitor central processor sends out a request for a given memory location and the interface to the memory retrieves the contents of that location when possible without disturbing the communication CPU. The interface flags the CPU when the data is ready and the monitor CPU reads in the data through an I/O type input port.

In the following two sections each of these techniques will be described in detail, including both their advantages and disadvantages

5.3 Time-Shared Bus Interface For Monitor CPU

As mentioned above, one technique for interfacing the monitor CPU to the two port RAM memory is to connect the monitor CPU to the standard memory bus and to time share the memory bus availaility

between the communication CPU and the monitor CPU. The normal operation of the central processor is controlled by a two phase clock as shown as 01 and 02 in Figure 1. The memory is accessed through tri-state buffers by

Anding the valid memory address line with the 02 line. In this way, during the 01 cycle all of the memory address and access lines are in the open tri-state configuration; therefore, a second CPU could freely access the memory if it is insured that all accesses were permitted only during the 01 cycle. This can be done by using one clock to control both the communication and the monitor CPU but reversing the phases on the monitor CPU, as illustrated in Figure 2.

However, this very simple clocking scheme produces certain timing problems because the 02 pulse is used as a memory strobe and the memory address lines may not have time to stablize prior to memory strobe. To eliminate these timing problems, a slightly more complex clocking scheme as shown in Figure 3 is used. In this scheme, the bus enable pulses that control which CPU can access the bus is separated from the 02 pulse to permit additional settling time between enablng the bus and the memory strobe. The schematic of the central processor before modification for the two port memory is shown in Figure 4. The central processor with the modifications are shown in Figure 5.

The synchronous clock waveforms for 01, 02 and the bus enable lines were implementd by blasting the on and off bit patterns into the first sixteen consecutive addresses of a read-only memory. The

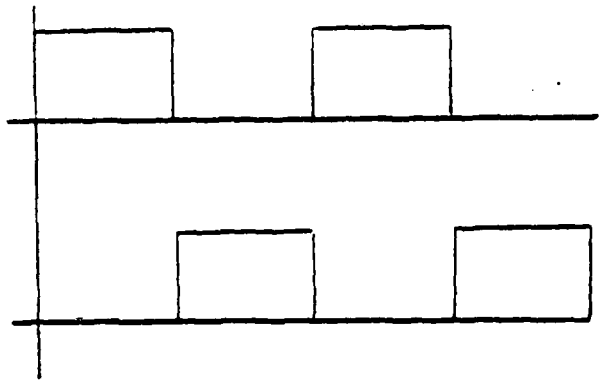


FIGURE 5.1 Two Phase Clock

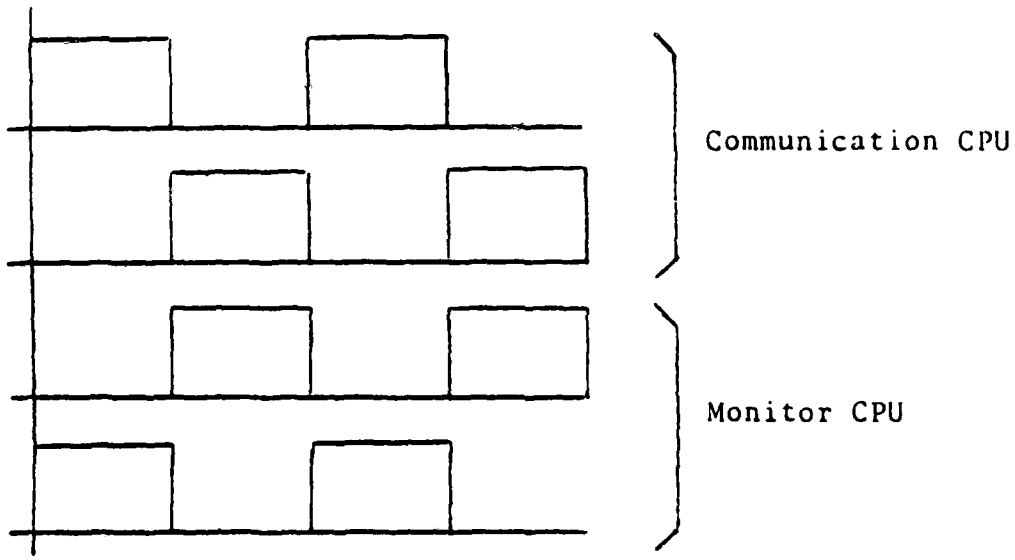


FIGURE 5.2 Dual CPU Clocks

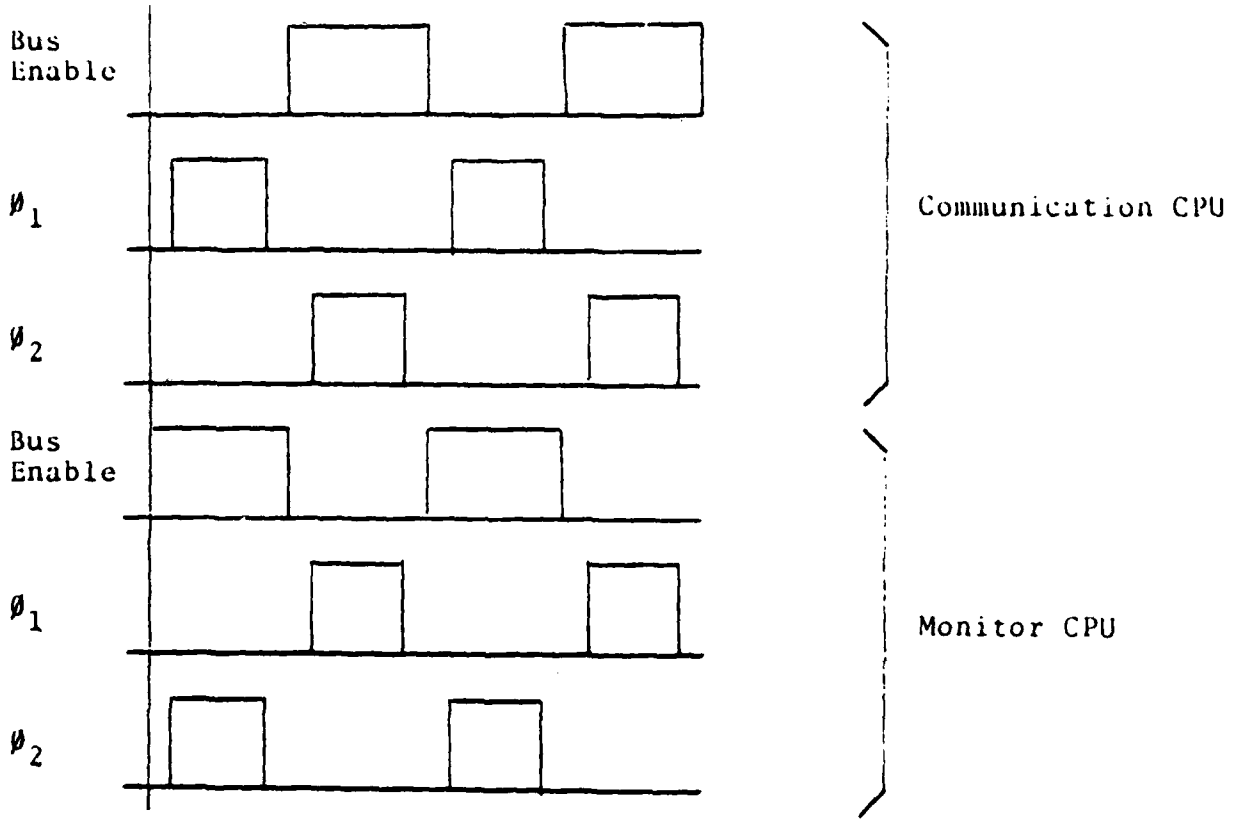


FIGURE 5.3 Modified Dual CPU Clock

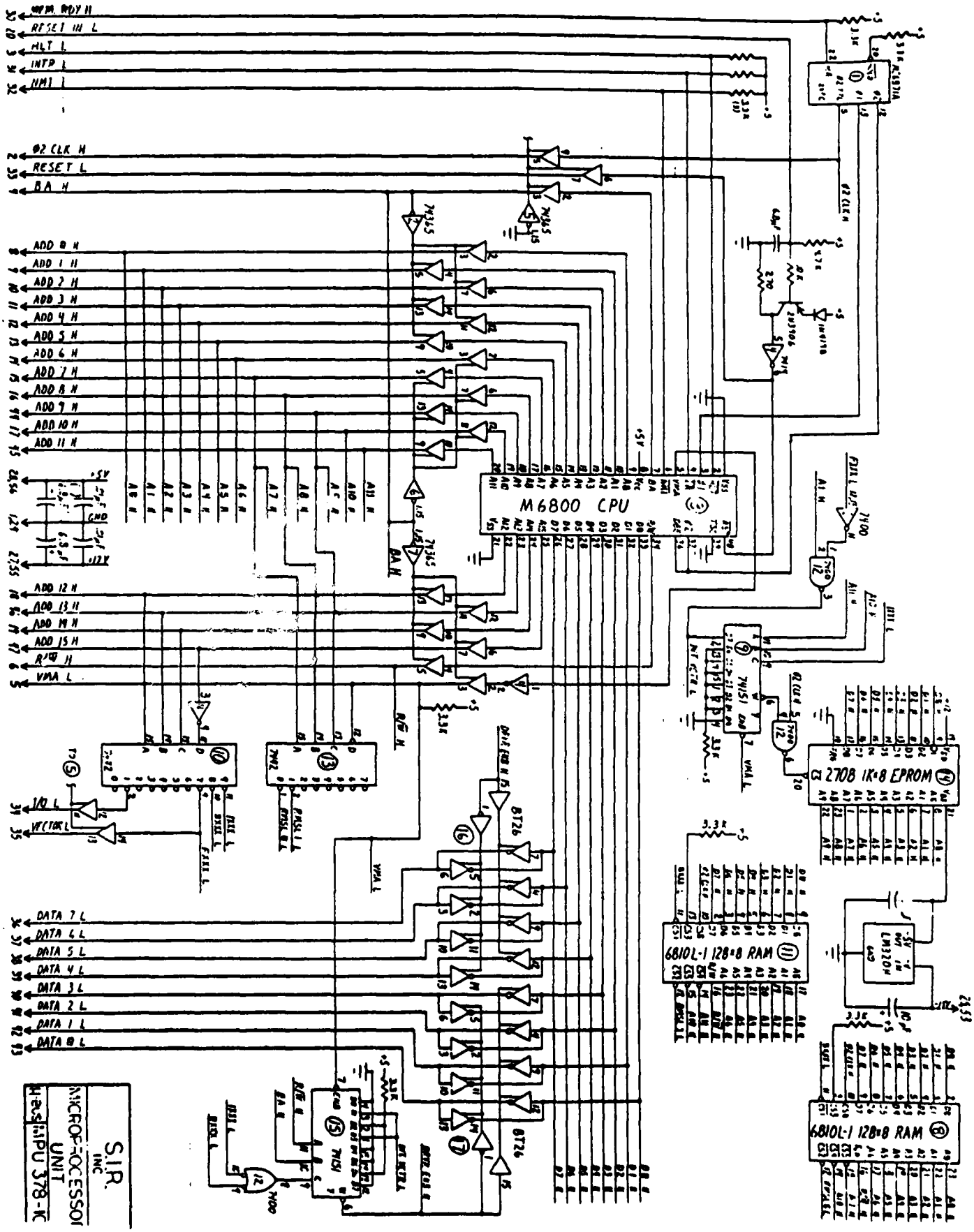


FIGURE 5.4 Central Processor

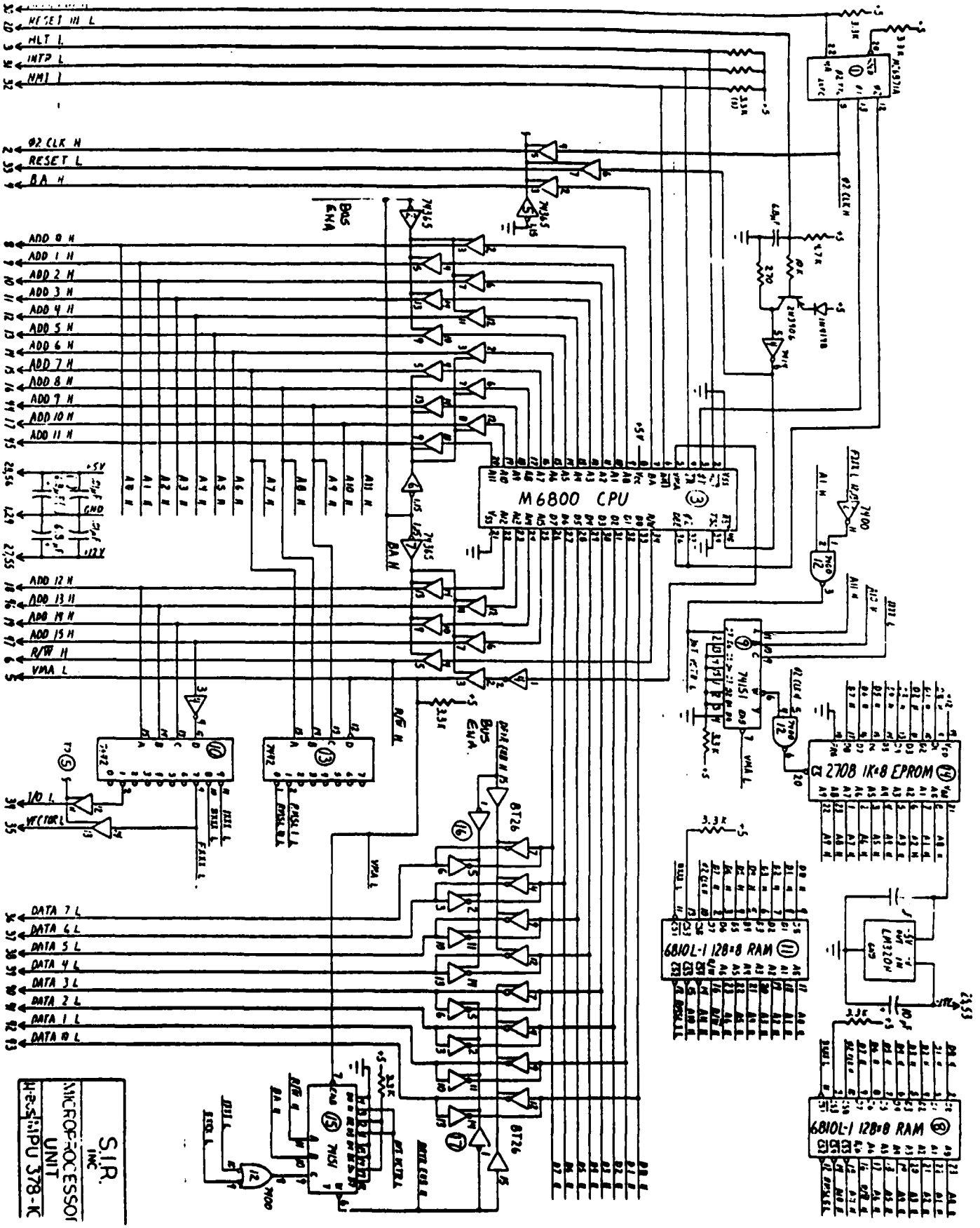


FIGURE 5.5 Modified Central Processor

ROM address lines are attached to a 16-bit counter which is continually clocked at sixteen times the effective 01, 02 clock rate. The ROM is therefore continually swept through the first sixteen addresses and produces all four of the waveforms needed for the CPU clocks. A functional block diagram of this clocking system is shown in Figure 6. This circuit was added to the system by constructing a new printed circuit board which contains the clock circuitry plus an additional RS232 serial port, which is used by the monitor CPU to transmit performance data. The circuit diagram of this card is shown in Figure 7.

The communication node hardware with performance monitoring is comprised of a communication CPU to control the communication process, a monitor CPU to control the monitoring process, a ROM memory for the communication and monitoring programs, a RAM memory for message data, four serial I/O ports of communication input and output, one serial port for performance data transmission and the time-sharing clock circuitry. These are distributed on printed circuit cards as illustrated in Figure 8.

5.4 I/O Based Interface For The Monitor CPU

As discussed in Section II, another possibility for interfacing the monitor CPU to the two-port memory is to interface the CPU by means of standard parallel I/O ports. With this technique, the monitor CPU sends out the address of the memory location it wants to read on sixteen I/O lines and sets a flip flop indicating that it is

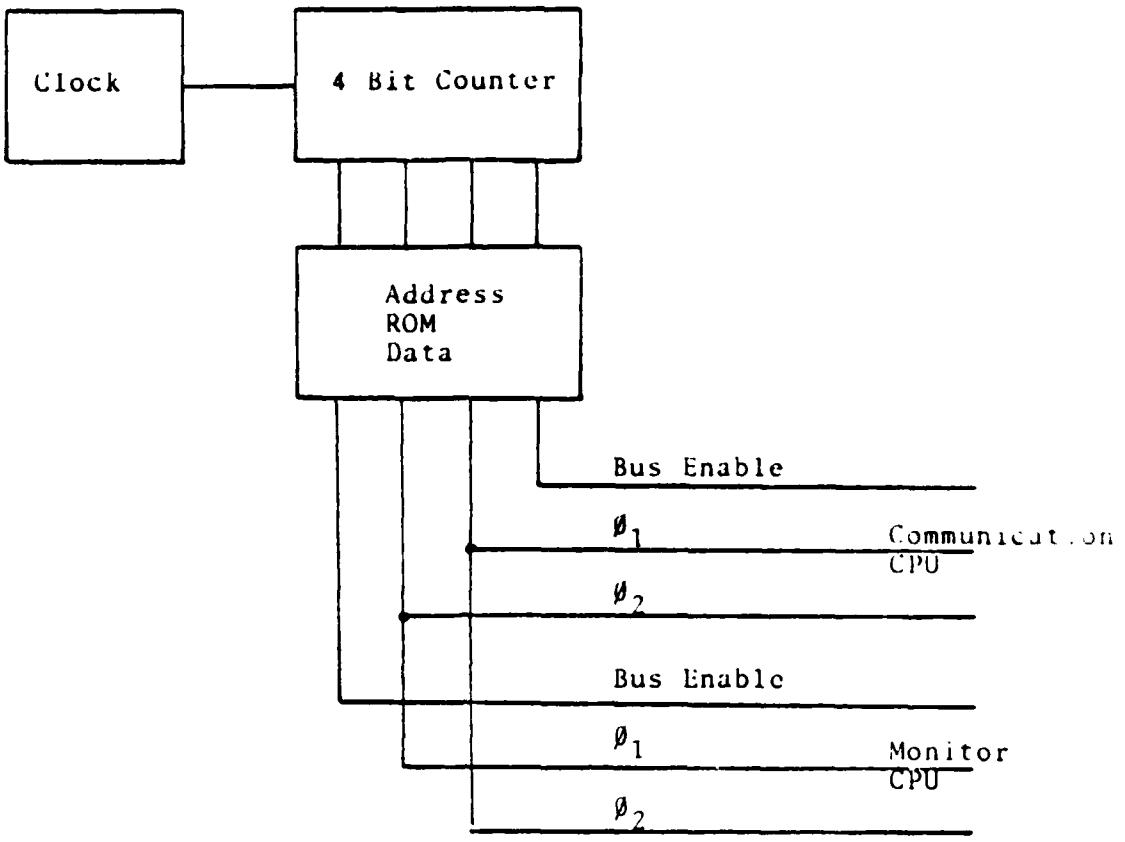


FIGURE 5.6 Clock Implementation

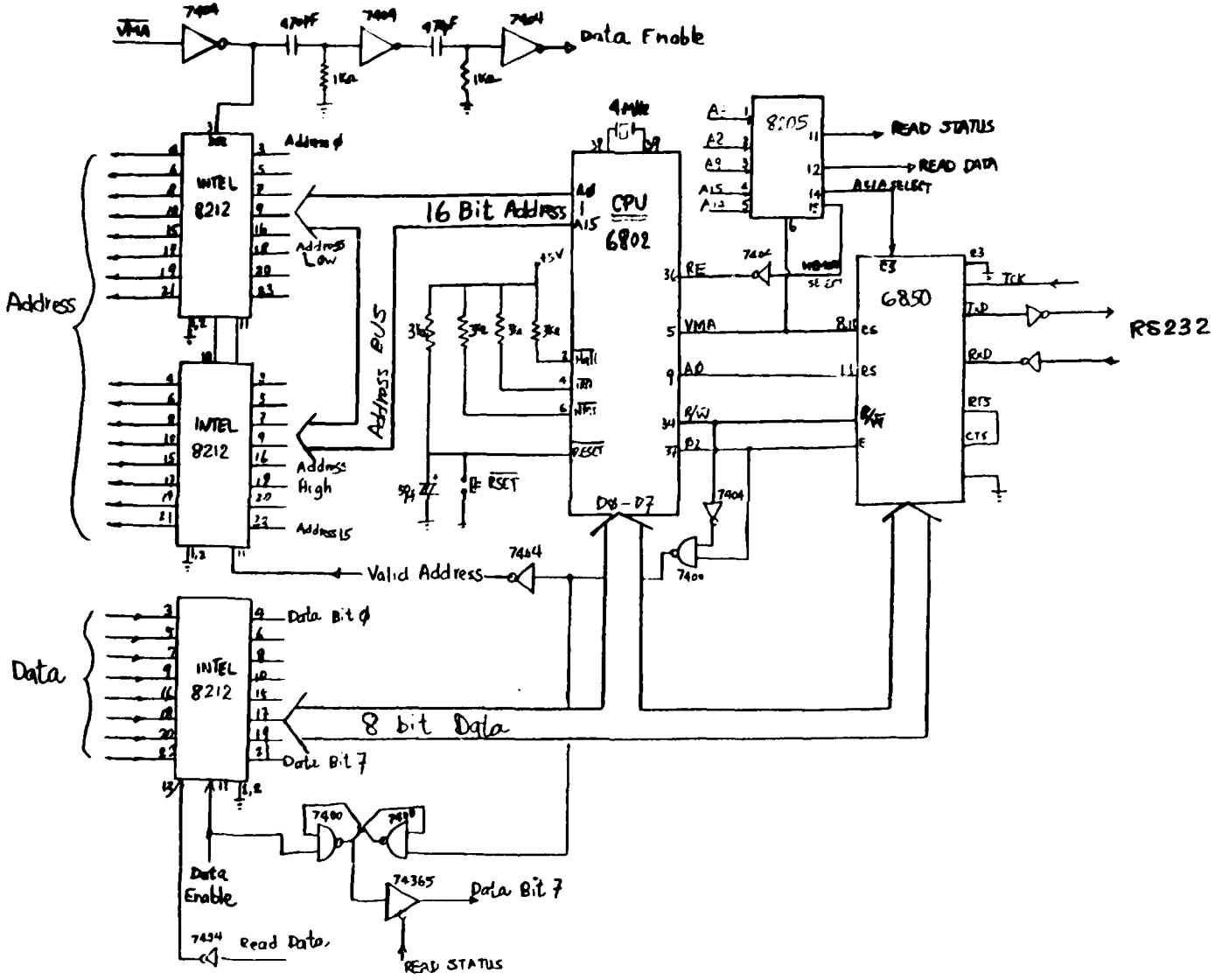


FIGURE 5.7 CIRCUIT DIAGRAM

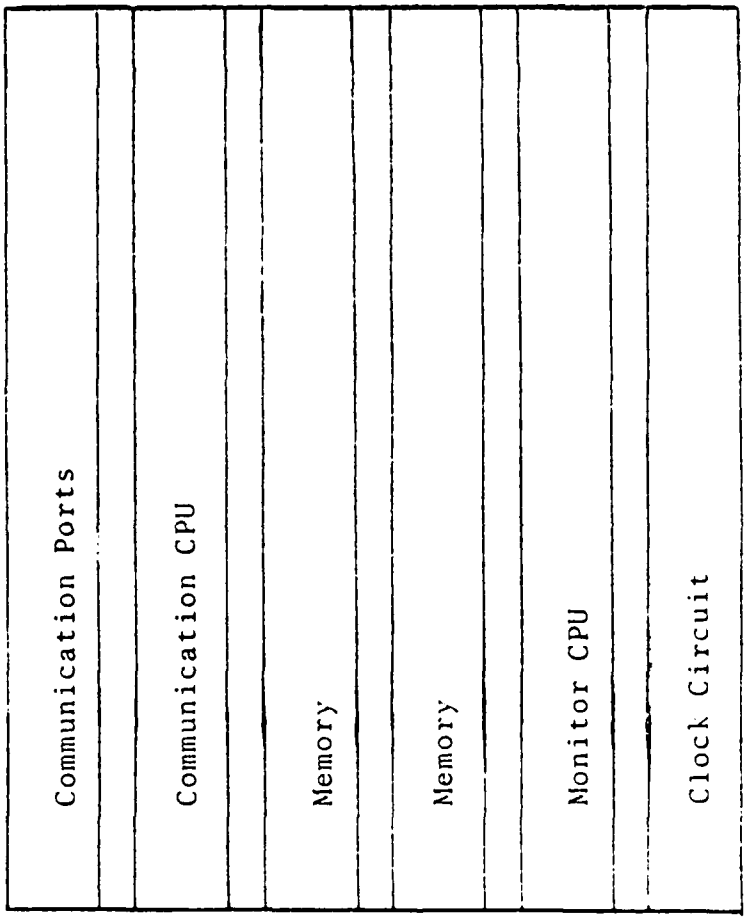


FIGURE 5.8 Card Layout

requesting data. Control circuitry interfaced to the memory bus then detects the next available time that the memory is not being accessed by the communication CPU, retrieves the contents of the specified memory location and places it in a holding buffer. The control logic re-sets the flag flip flop as an indication to the monitor CPU that the data has been retrieved. The monitor CPU then reads the holding register using normal parallel I/O lines. A functional block diagram of this system is shown in Figure 9.

A prototype system using a 6502 microprocessor as shown in Figure 10 was built and successfully operated with the communication 6800 CPU and memory board.

5.5 Comparison of the Time-Sharing Interface and I/O Interface Techniques

Both the time-shared interface and the I/O interface for the monitor CP were successfully built and tested with the communication 6800 central processor and two port memory. The advantages and disadvantages of these techniques will influence the selection of which technique will be used for a particular application. The following is a list indicating the advantages of each.

Time-Shared Bus Interface Advantages:

1. Very little specialized hardware is required for implementation.

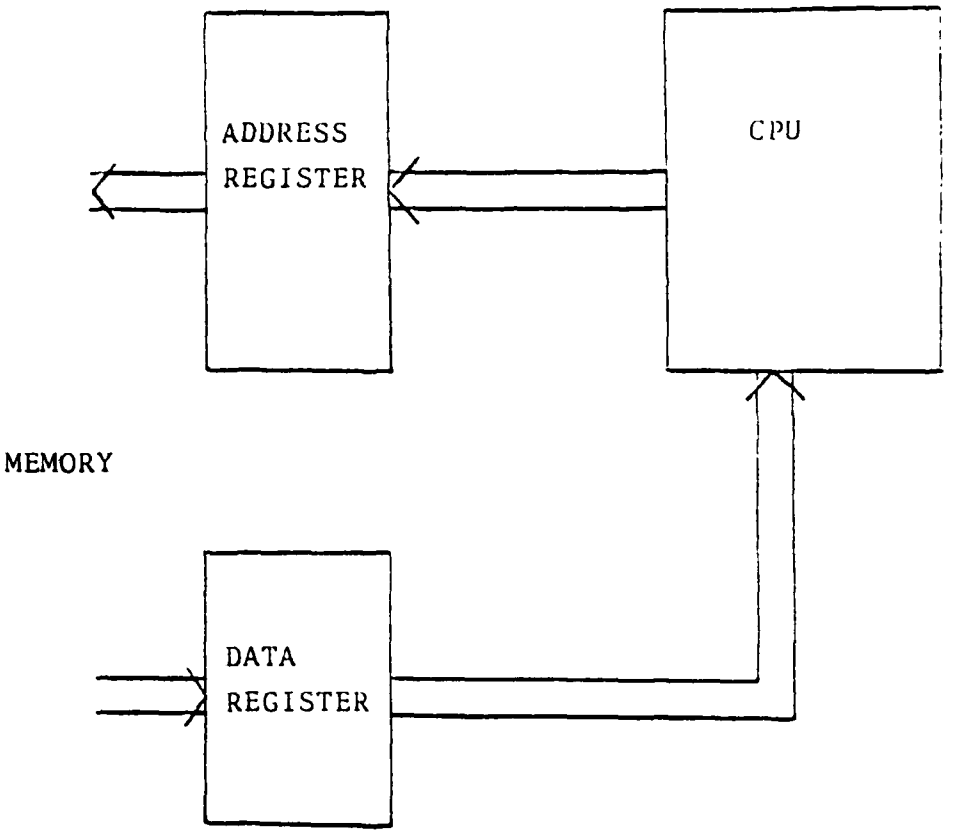


FIGURE 5.9 Prototype System Block Diagram

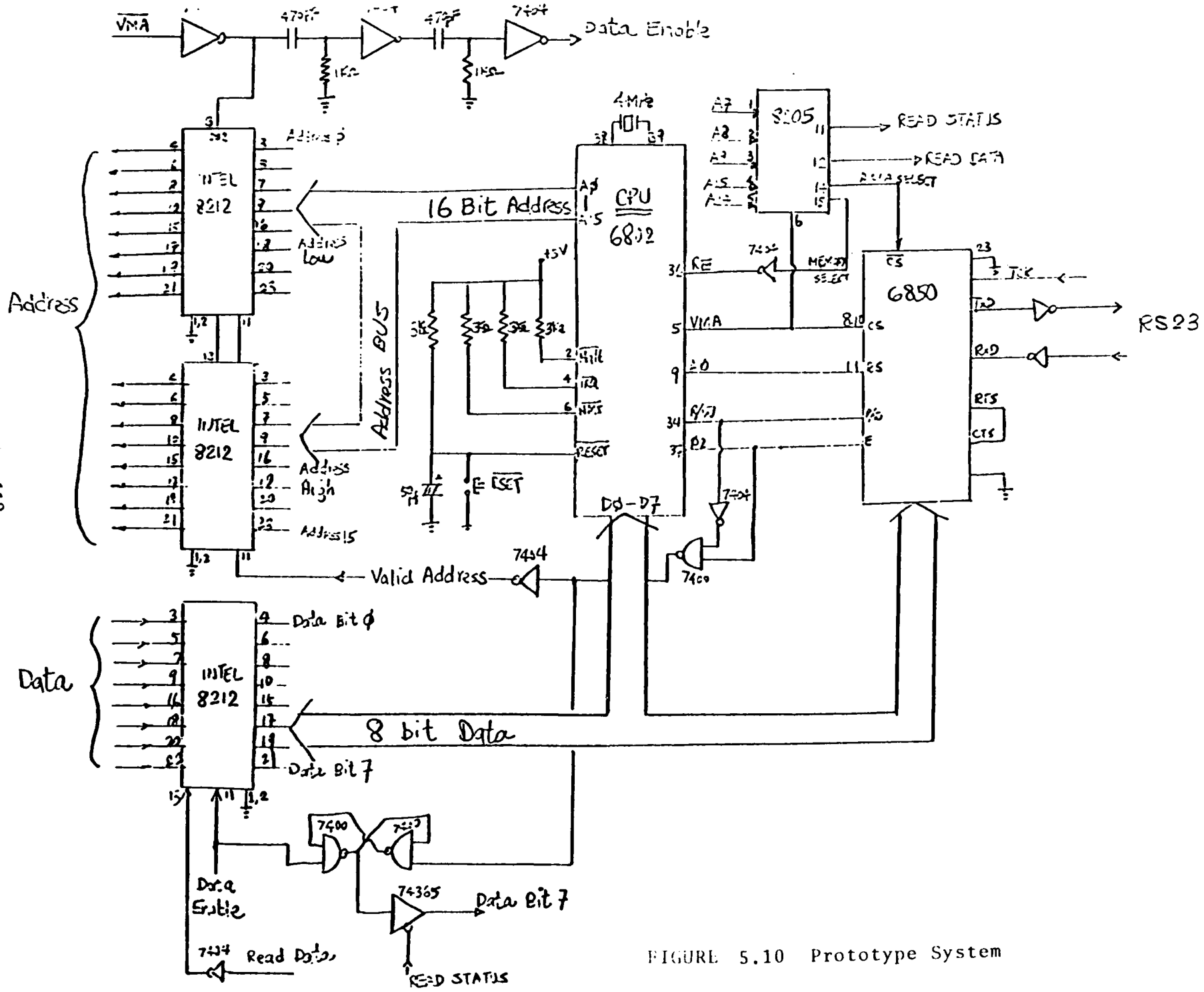


FIGURE 5.10 Prototype System

2. The monitor CPU access to the two port memory is extremely fast since it never has to wait on the communication CPU for access. Software in the monitor CPU can be minimized since a single instruction is adequate for reading the two port memory.

I/O Interface Advantages

1. Does not require as fast a memory as the time shared techniques.

2. Can be used between different types of CPU.

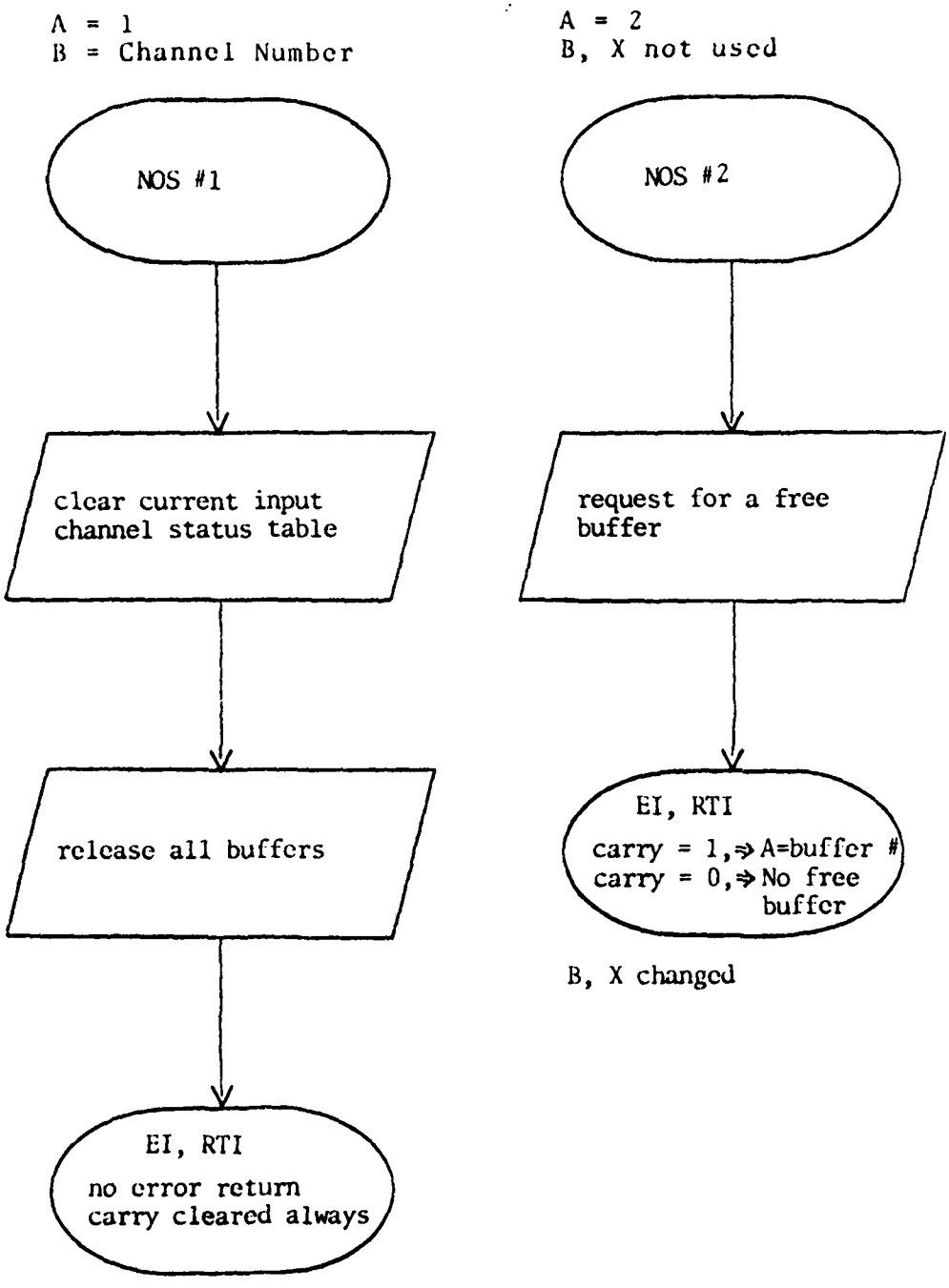
APPENDIX A

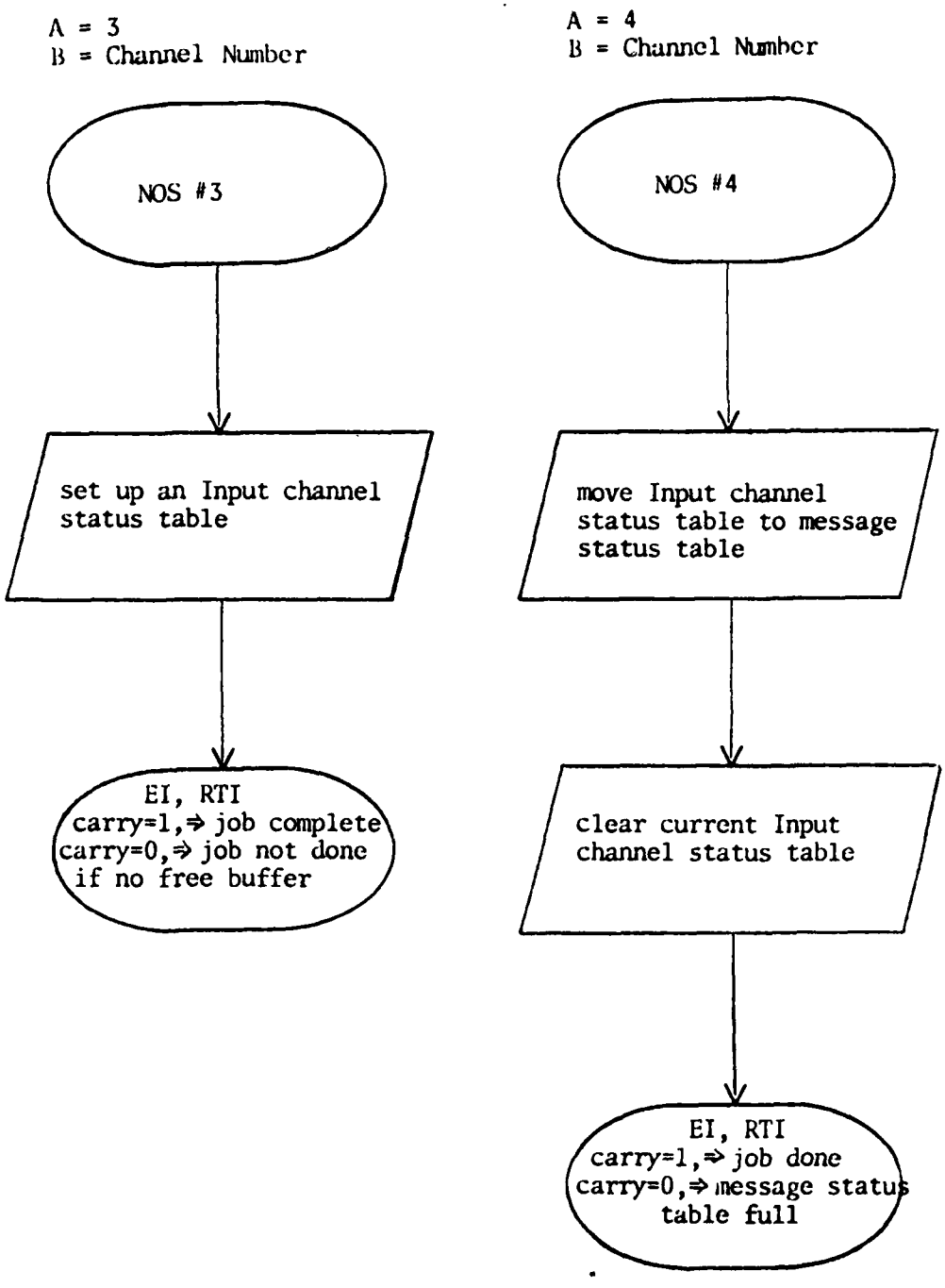
Communication Software Flow Charts

BUFFER STATUS TABLE

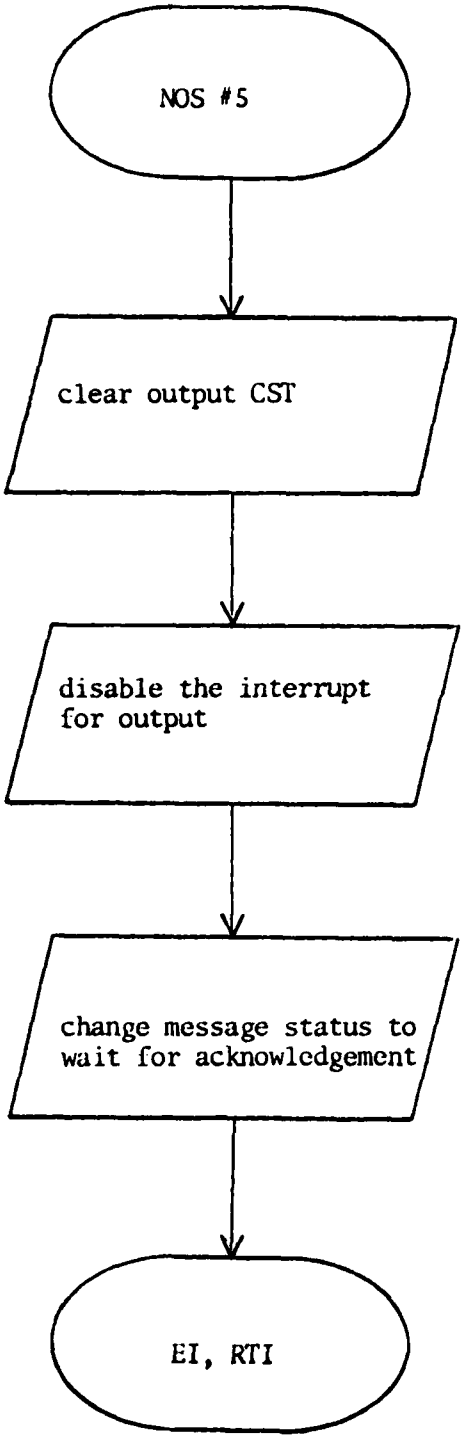
Relative Location	Name	Description
0	BST	-One byte per buffer -Buffer is a non-zero number -0 = Not in use -FF = In use by CST -FE = Local acknowledgement packet -FD = Source acknowledgement packet -FC = Reserved for monitor -ELSE = Message number
.		.
.		.
.		.
.		.
NOBUF		.

B) FLOW CHART OF NOS SUBROUTINE CALLS

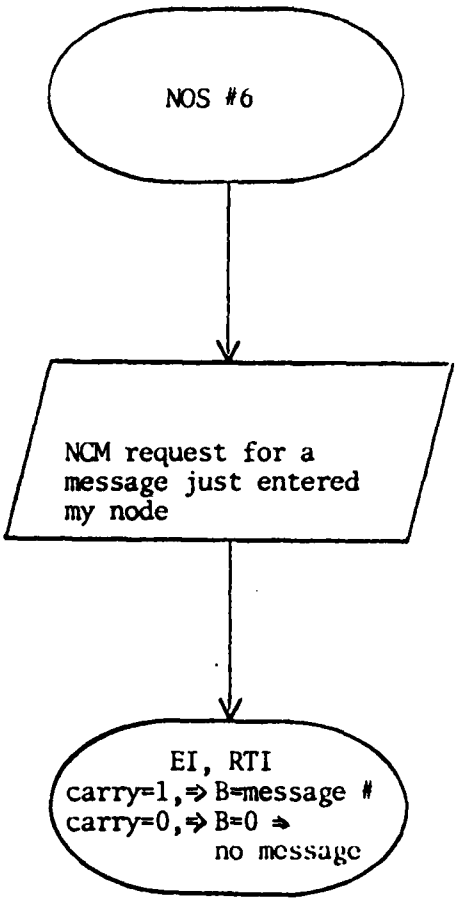


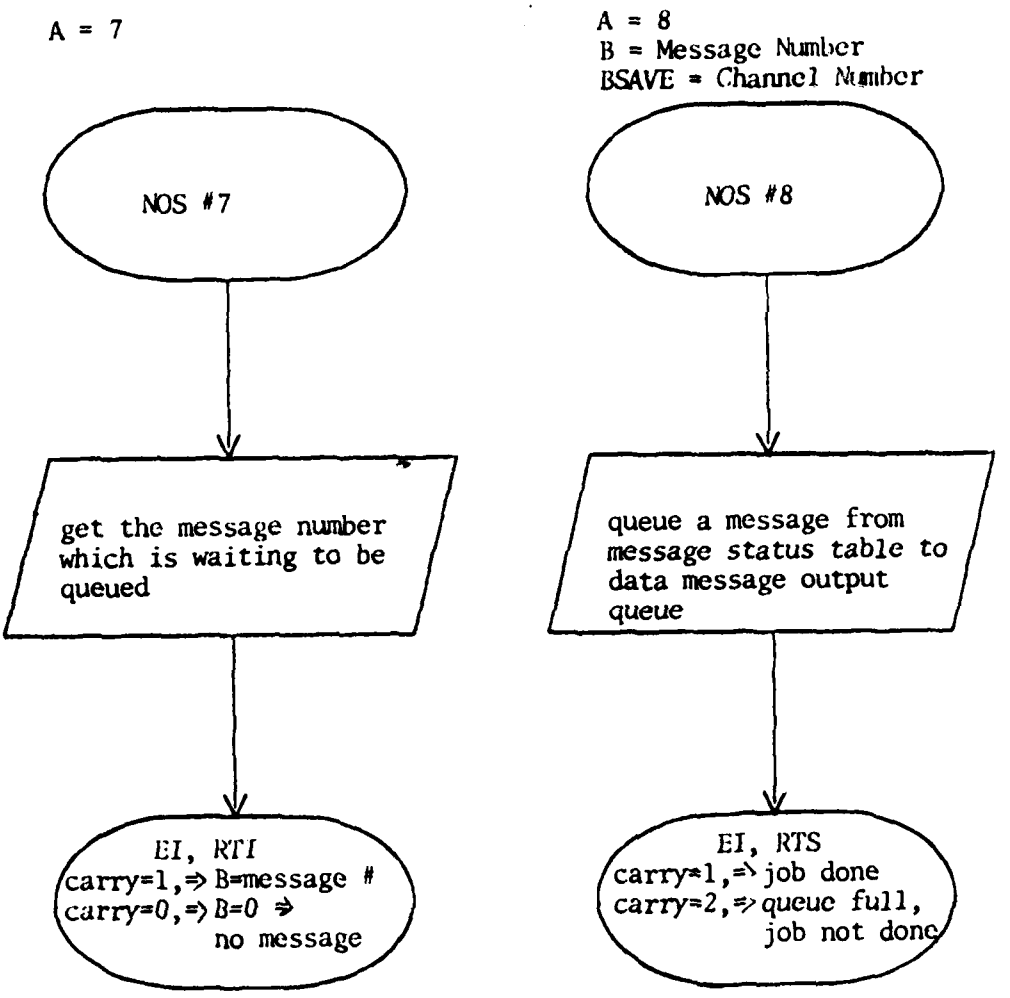

A = 3
B = Channel Number
NOS #3
set up an Input channel status table
EI, RTI
carry=1,⇒ job complete
carry=0,⇒ job not done
if no free buffer
A = 4
B = Channel Number
NOS #4
move Input channel status table to message status table
clear current Input channel status table
EI, RTI
carry=1,⇒ job done
carry=0,⇒ message status table full

A = 5
B = Channel Number

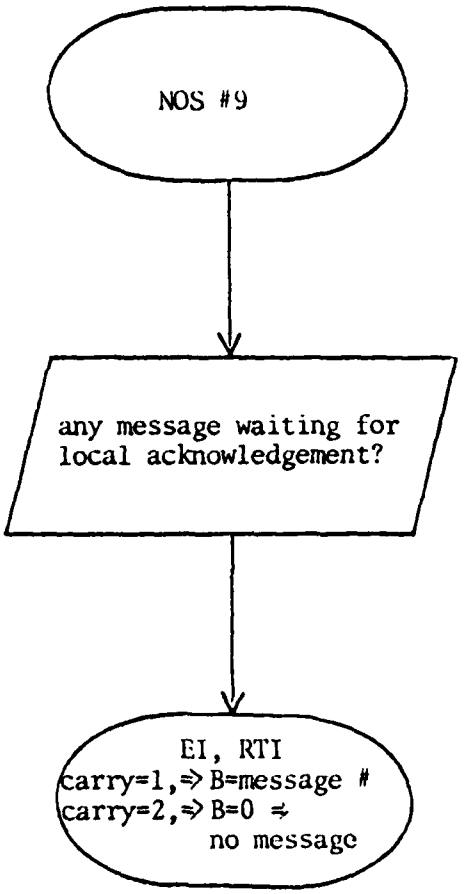


A = 6

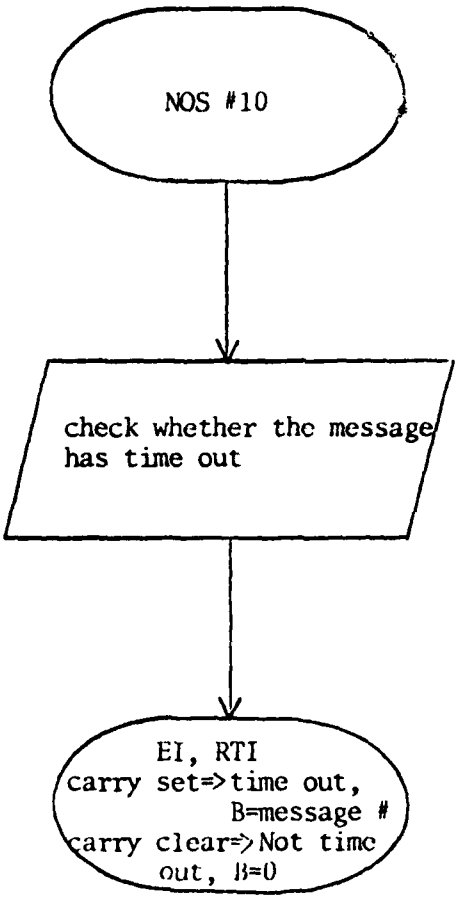


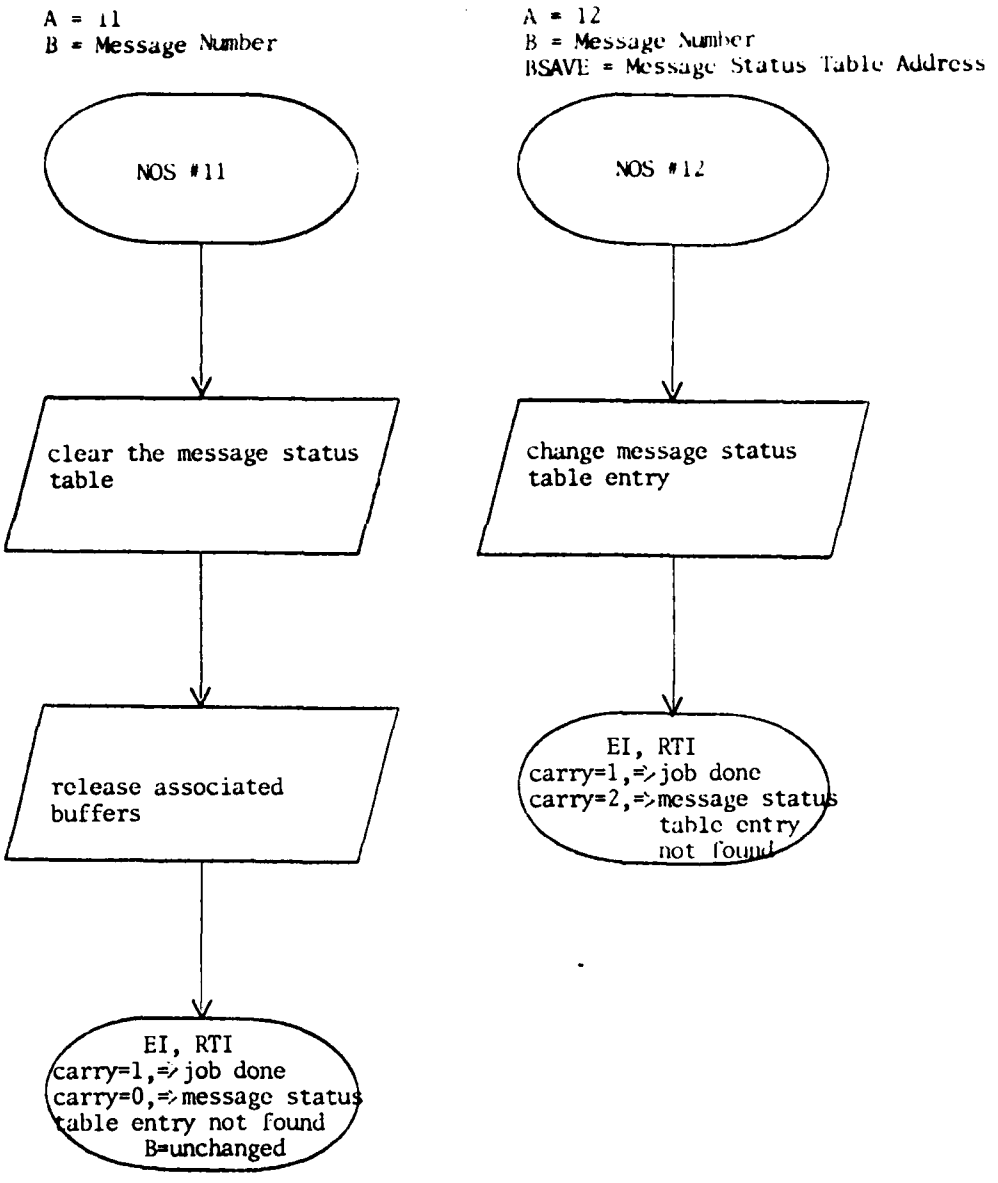

A = 7
NOS #7
get the message number which is waiting to be queued
EI, RTI
carry=1, ⇒ B=message #
carry=0, ⇒ B=0 ⇒ no message
A = 8
B = Message Number
BSAVE = Channel Number
NOS #8
queue a message from message status table to data message output queue
EI, RTS
carry=1, ⇒ job done
carry=2, ⇒ queue full, job not done

A = 9

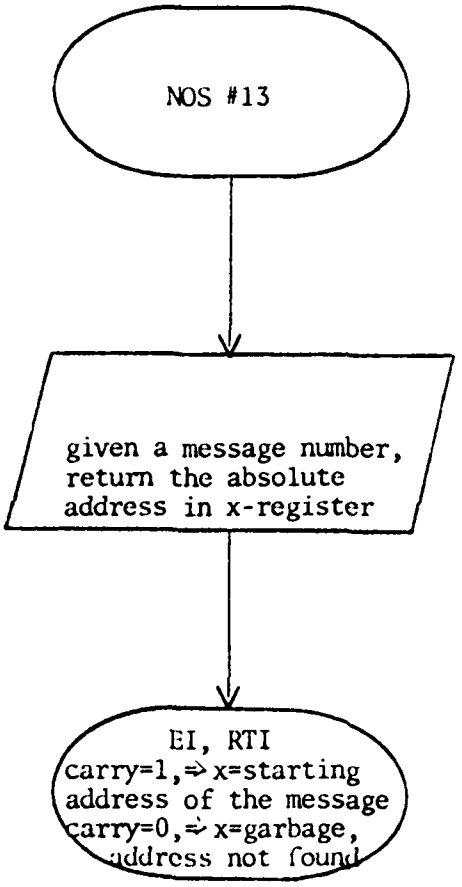

NOS #9
any message waiting for local acknowledgement?
EI, RTI
carry=1,⇒ B=message #
carry=2,⇒ B=0 ⇒
no message

A = 10

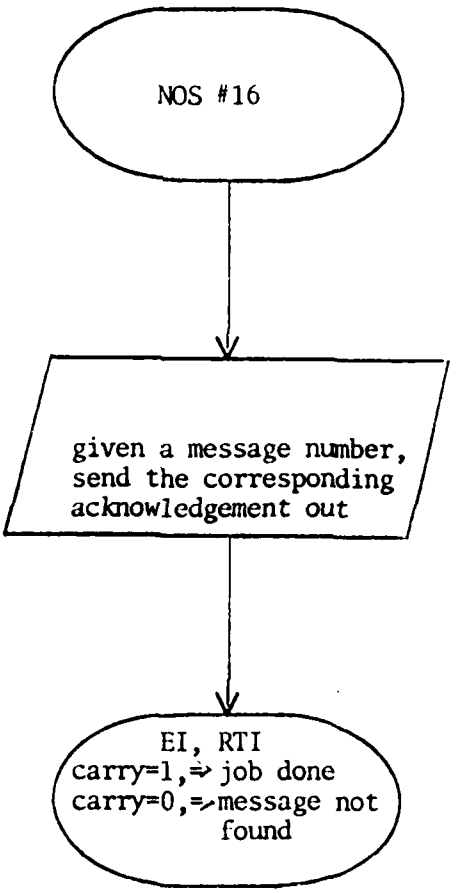

NOS #10
check whether the message has time out
EI, RTI
carry set⇒time out,
B=message #
carry clear⇒Not time
out, B=0

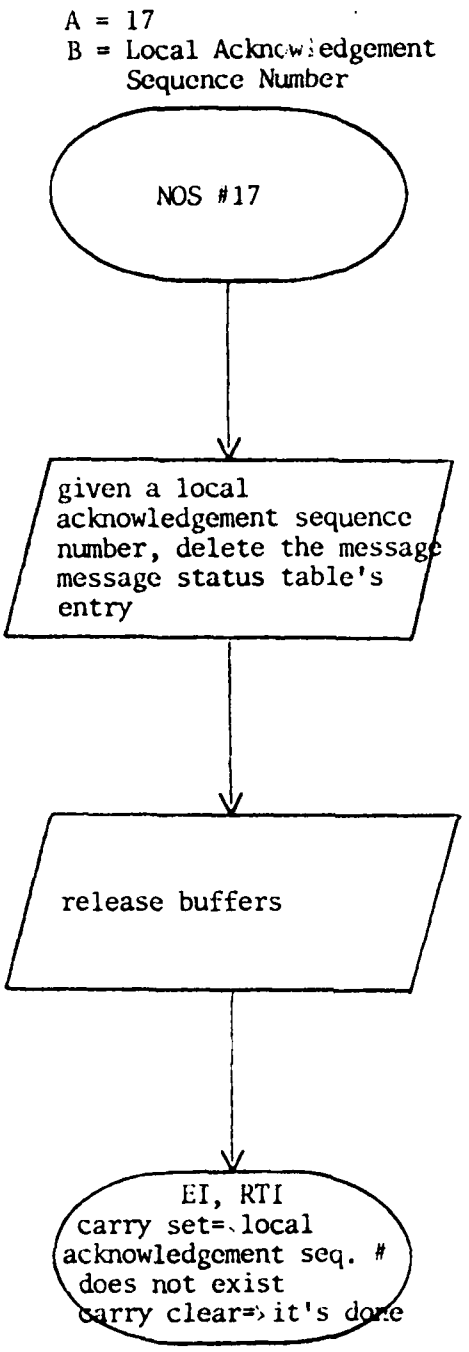

A = 11
B = Message Number
NOS #11
clear the message status table
release associated buffers
EI, RTI
carry=1,=>job done
carry=0,=>message status table entry not found
B=unchanged
A = 12
B = Message Number
BSAVE = Message Status Table Address
NOS #12
change message status table entry
EI, RTI
carry=1,=>job done
carry=2,=>message status table entry not found

A = 13
B = Message Number



A = 16
B = Message Number




A = 17
B = Local Acknowledgement Sequence Number
NOS #17
given a local acknowledgement sequence number, delete the message message status table's entry
release buffers
EI, RTI
carry set=>local acknowledgement seq. # does not exist
carry clear=>it's done

CHANNEL STATUS TABLE DEFINITION

Input:

0) 1 Byte - First Buffer Number (Non-Zero)

1) 1 Byte - Total Number of Buffers

2) 1 Byte - Present Input Buffer Number

3) 1 Byte - Location in Present Buffer

4) 1 Byte - E-Flag, 0 = Not Found, 1 = Found

5) 1 Byte - High Order LRC (Real)

6) 1 Byte - Low Order LRC (Real)

7) 1 Byte - High Order LRC (Temporary)

8) 1 Byte - Low Order LRC (Temporary)

9) 1 Byte - High/Low LRC Switch, 0 = High, 1 = Low

Output:

0) 1 Byte - Total Number of Output Buffer Left
1 = Last Buffer, 0 = No Output Channel Status Table

1) 1 Byte - Number of Bytes in the Last Buffer

2) 1 Byte - Present Buffer Number

3) 1 Byte - Location in Present Buffer

4) 1 Byte - E-Flag

5) 1 Byte - STX-Flag, 1 = STX Sent, 0 = Not Sent

6) 1 Byte - EXT-Flag, 1 = ETX Sent, 0 = Not Sent

7) 1 Byte - Output LRC High (Not Used)

8) 1 Byte - Output LRC Low

9) 1 Byte - High/Low Switch for LRC

A) 2 Bytes- Address of This Message Status in MST

C) 1 Byte - Channel Sequence Number

PACKET FORMAT

Relative Position	Content	Description
00	02	STX (Start of message)
01	--	Packet type identifier
	01	Source acknowledgement
	02	Local acknowledgement
	03	Data Packet
	04	Executable data packet
02	1<x<$32	Total number of buffers needed for this packet
03	0<x< maximum buffer size	Number of bytes in the last data packet
04	x	Packet origin
05	x	Packet destination
06	x	Origin message sequence number
07	x	Origin packet sequence number
08	x	Local packet sequence number
09	x	Data
0A	x	.
.	x	.
.	x	.
y	x	Data
y + 1	x	High order LRC
y + 2	x	Low order LRC
ETX	04	End of packet

y = Maximum buffer size -3

MESSAGE STATUS TABLE DEFINITION

0) 1 Byte - Processing Status:

- 0 = No message (this entry not in use)
- 1 = This waiting packet has been processed and is to be queued to the channel specified in location 8
- 2 = Packet received without any detectable LRC error
- 3 = This packet is waiting for local acknowledgement
- 4 = This packet is waiting for source acknowledgement
- 5 = This is a local acknowledgement packet (highest priority)
- 6 = This packet is waiting for output; after outputting the PS should change to 3
- 10 = Data packet for my node
- 11 = Local acknowledgement has sent for data packet for my node
- 12 = Local acknowledgement for my node
- 1C = Local acknowledgement packet received, but the corresponding waiting for local acknowledgement packet not found
- 13 = Source acknowledgement for my node
- 14 = Data packet not found
- 15 = Local acknowledgement has sent for PS 14
- 20 = Packet not for my node
- FF = The MST entry should be cleared and associated buffers should be released

1) 1 Byte - Local Channel Sequence Number

2) 1 Byte - First Buffer Number

3) 1 Byte - Total Number of Buffers

4) 1 Byte - Total Number of Words in the Last Buffer

5) 2 Bytes- Five Minute Timer for Retransmission - Use Only Lower Byte

7) 1 Byte - Total Number of Times Retransmitted

8) 1 Byte - Channel Number (1-4) for Output

9) 1 Byte - Input Channel Number

A) 2 Bytes- Starting Address of the Buffer After Processed by PS 2

NETWORK OPERATING SYSTEM SUBROUTINE CALLS

A) Convention

Input:

1) A - Register A (always has the subroutine call identifier)

2) Interrupt always disabled

3) B - Register B

4) X - Index Register X

5) RTI - EXIT interrupt routine

6) EI - Enable interrupt

7) Channel Numbers are from one to four (1 - 4)

Output:

A, B, and X registers may or may not be changed, depending on each subroutine call. For details, please see the following assembler listing.

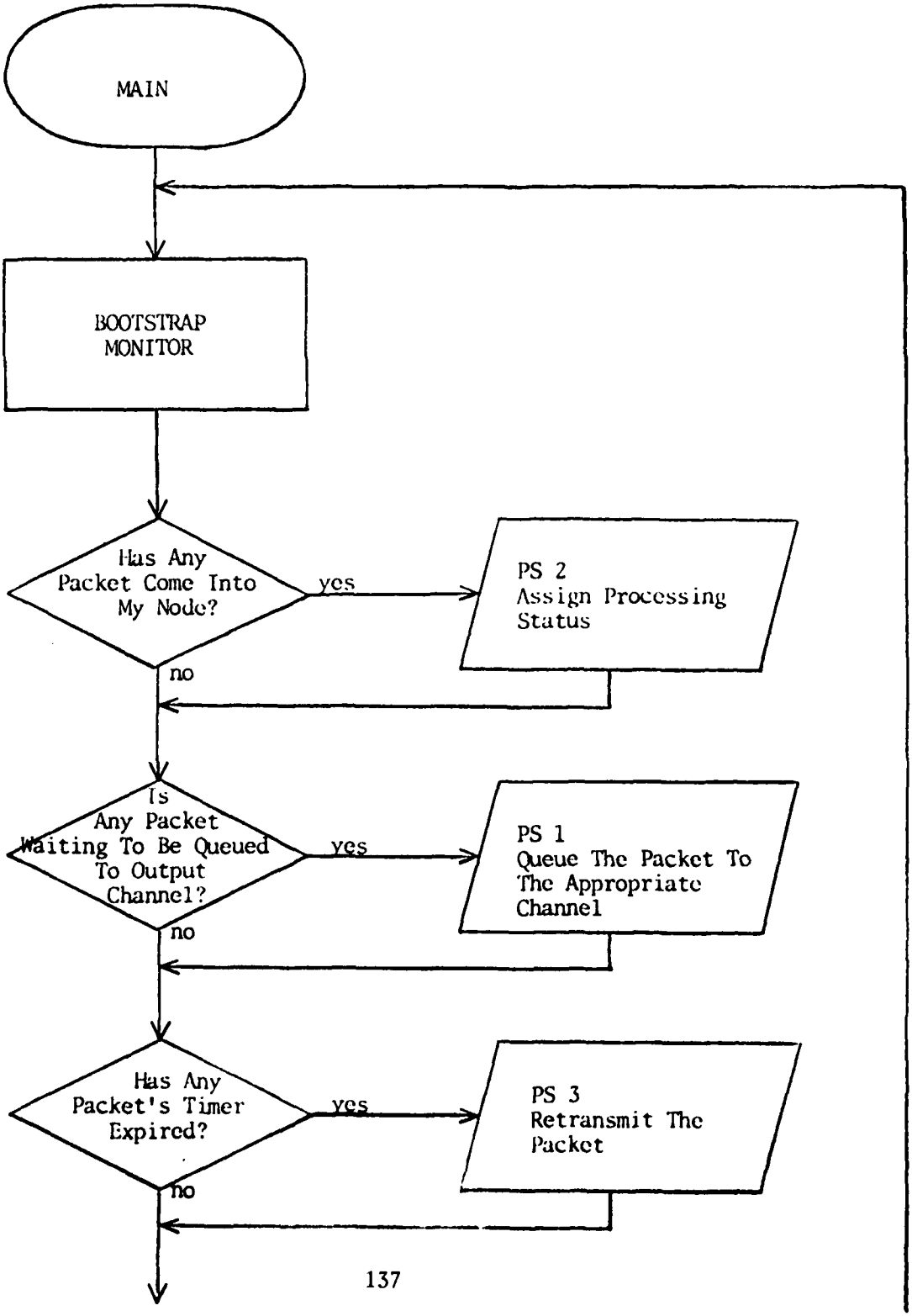
VIRTUAL MONITOR REGISTERS

Relative Location	Name	Description
0	CTSC1	Channel table sequence counter for channel 1
1	CTSC2	Channel table sequence counter for channel 2
2	CTSC3	Channel table sequence counter for channel 3
3	CTSC4	Channel table sequence counter for channel 4
4	ICUC1	Input channel utilization counter for channel 1
5	ICUC2	Input channel utilization counter for channel 2
6	ICUC3	Input channel utilization counter for channel 3
7	ICUC4	Input channel utilization counter for channel 4
8	OCUC1	Output channel utilization counter for channel 1
9	OCUC2	Output channel utilization counter for channel 2
10	OCUC3	Output channel utilization counter for channel 3
11	OCUC4	Output channel utilization counter for channel 4
12	MPUID	MPU idle count
13	MSC	Message sequence counter

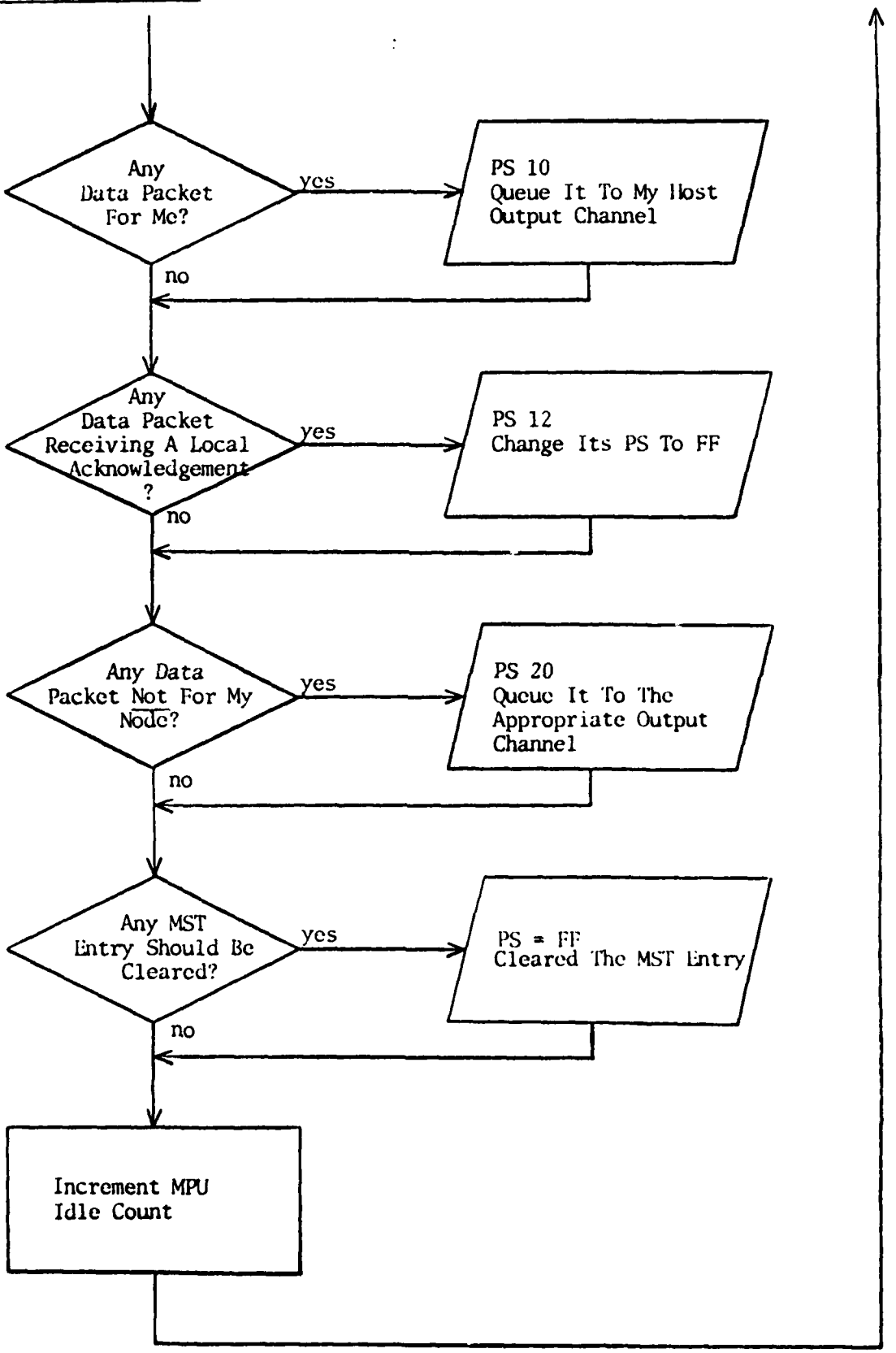
VIRTUAL MONITOR ERROR REGISTERS

Relative Location	Name	Description
0	E1	Unrecognizable interrupt
1	E2	ACIA input hardware error
2	E3	Input CST not found, and the incoming character is not STX
3	E4	Buffer full (incoming message will be dropped)
4	E5	ACIA output hardware error
5	E6	Message too long & not enough buffer to store it
6	E7	LRC error for both control & data packet
7	E8	Unrecognizable control packet
8	E9	Message needs to be retransmitted
9	E10	Messages need to be retransmitted
10	E11	Number of local acknowledgements received
11	E12	Interrupt asserted but neither input nor output
12	E13	Local acknowledgement does not exist
13	E14	Unrecognizable NOS command
14	E15	Release wrong buffer
15	E16	MST full, message tossed
16	E17	Output portion of CST does not exit
17	E19	Transmitter interrupted but TBUSY is not equal to 1
18	E20	Unknown packet type
19	E21	Status of bad ACIA
20	BACIA	Channel number of bad ACIA
21	BACIA+1	1 = input; 0 = output

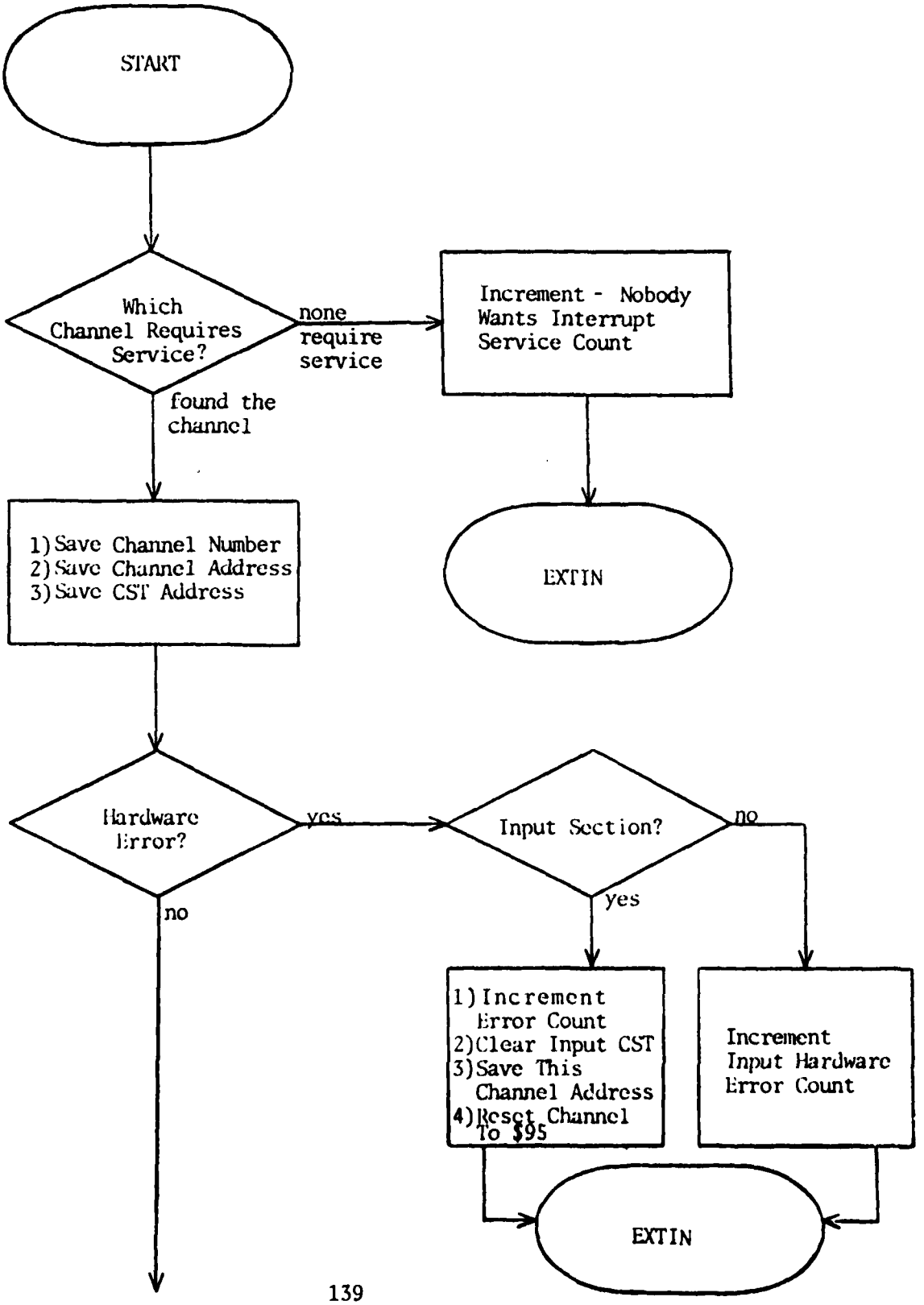
NETWORK MONITOR



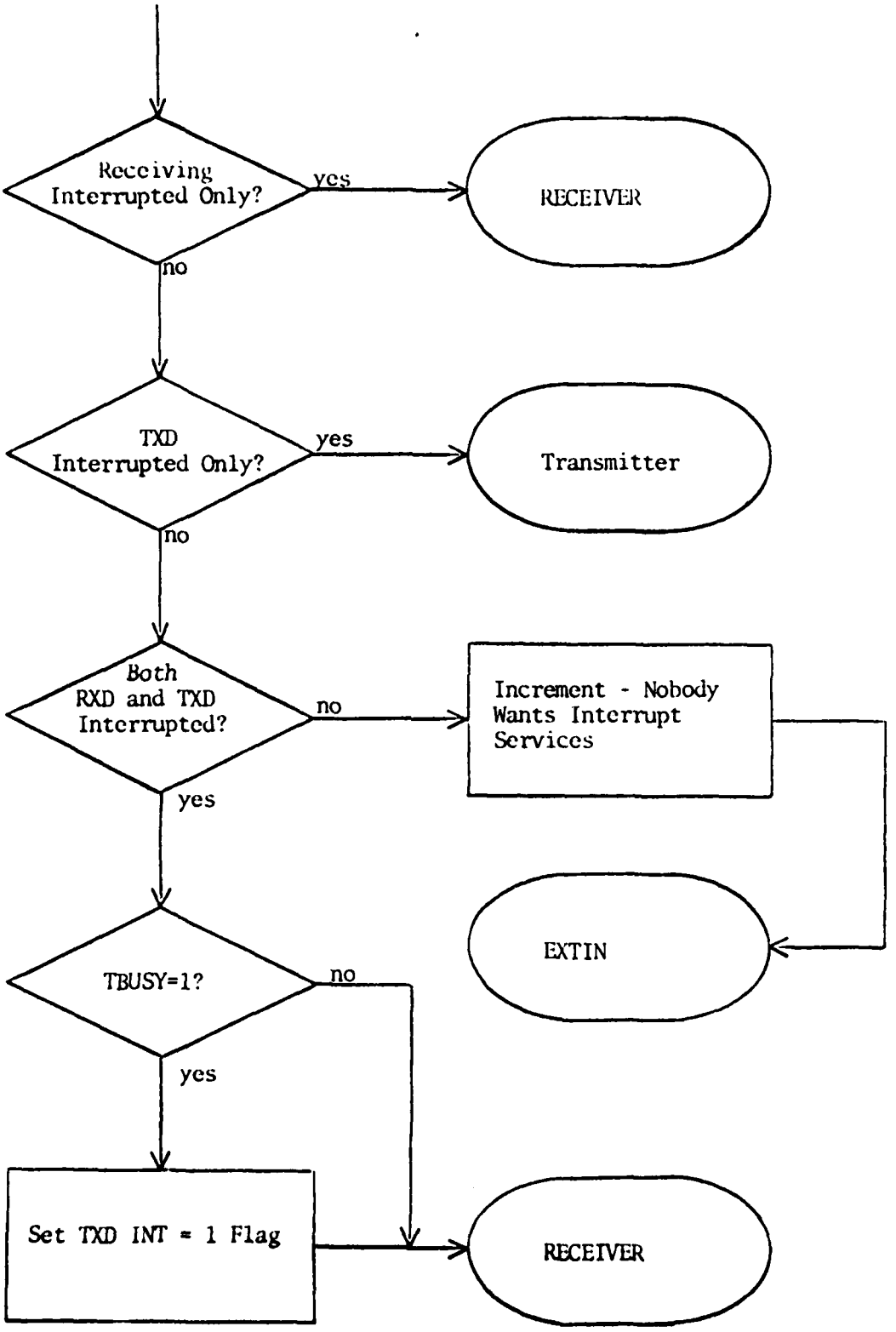
NETWORK MONITOR, con't



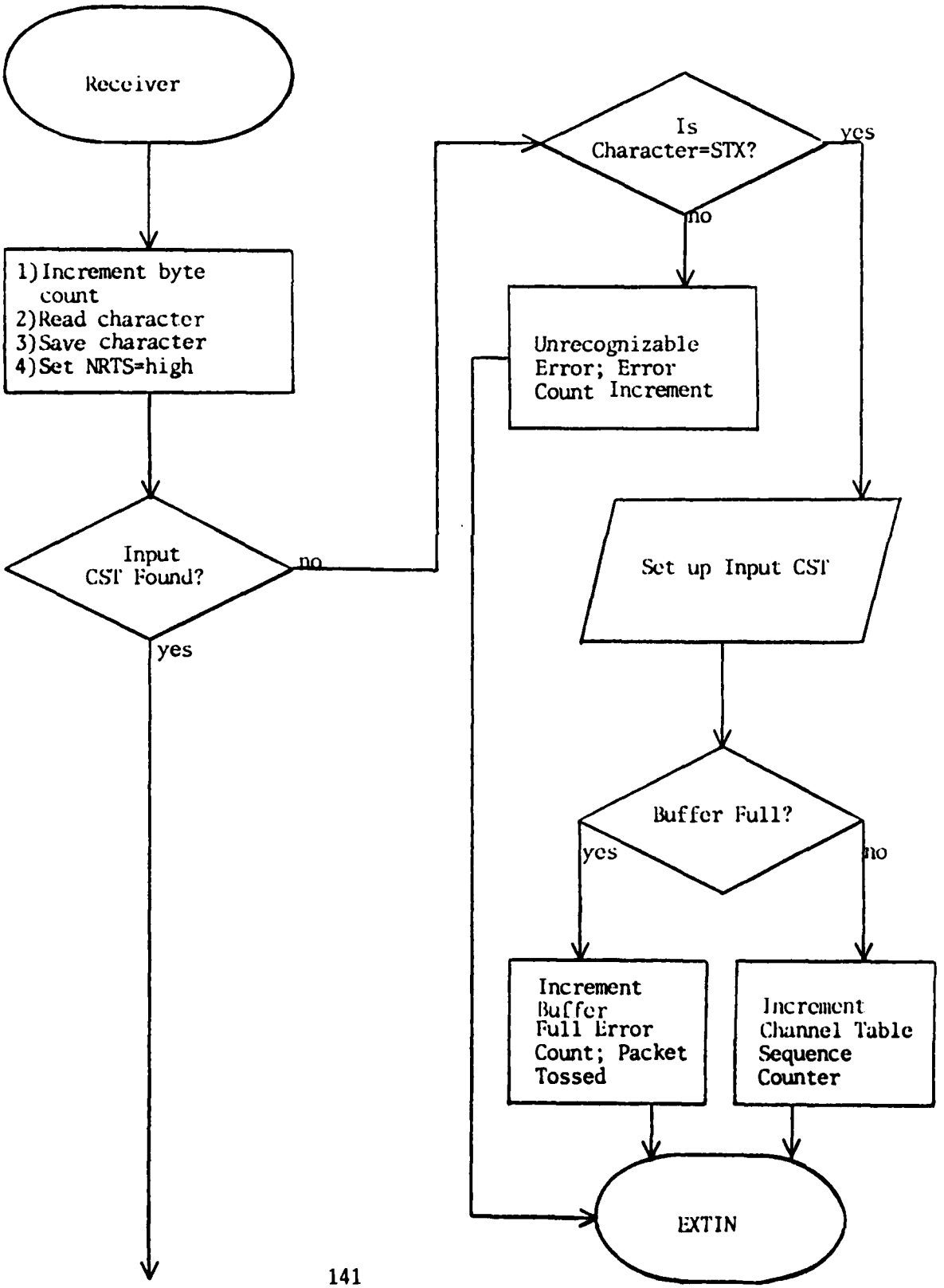
RE-ENTRANT INTERRUPT DISPATCH ROUTINE



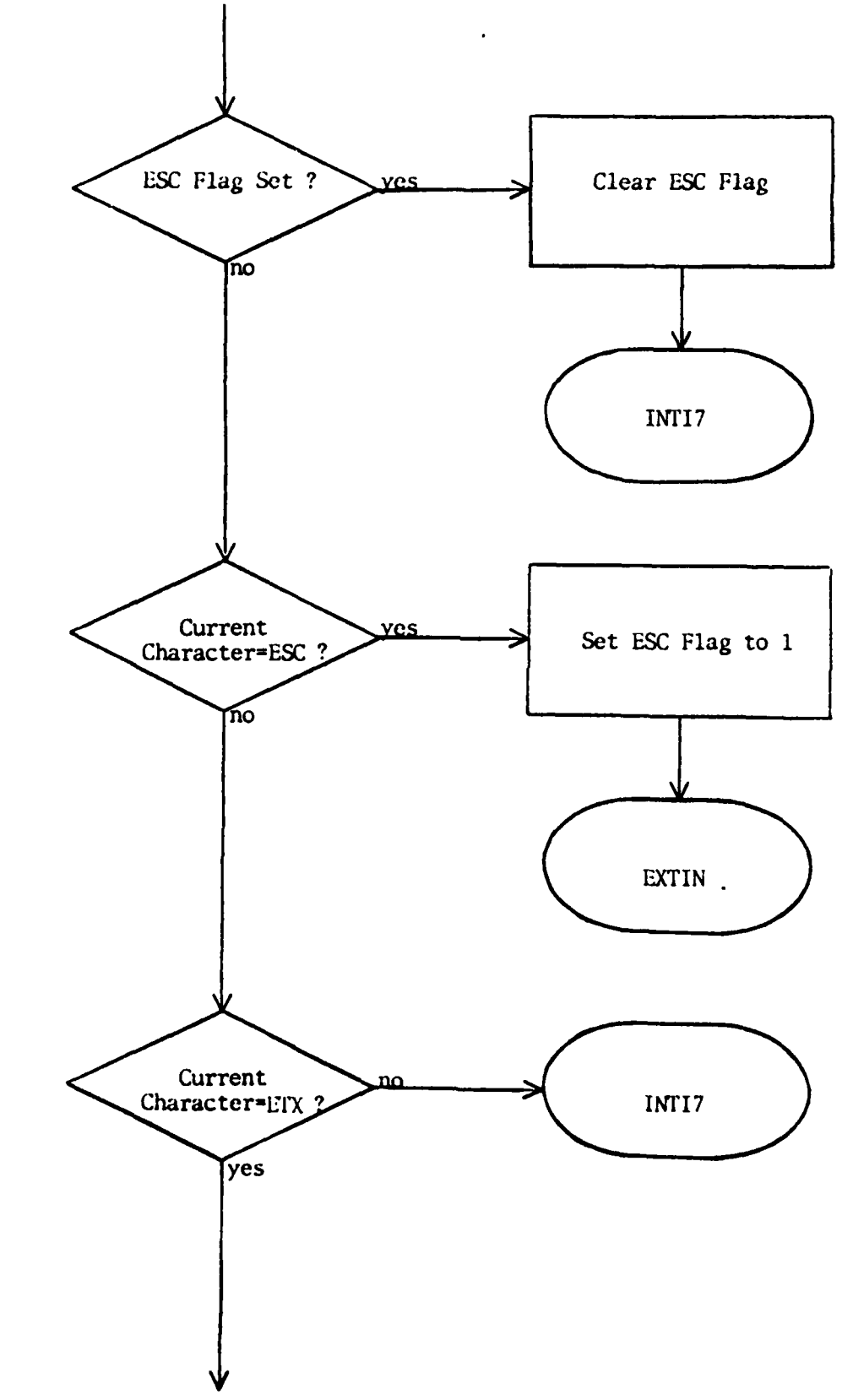
Re-entrant Interrupt Dispatch Routine, con't



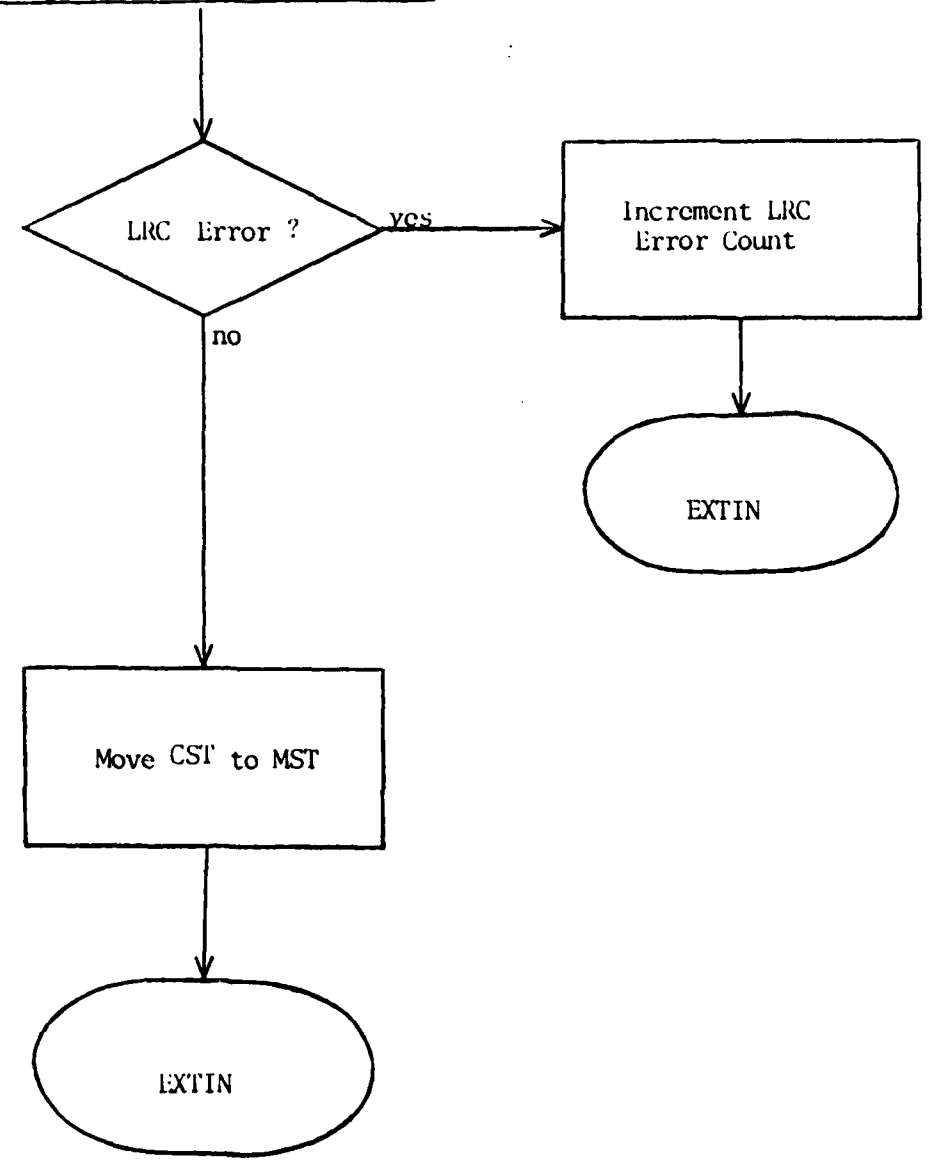
INTERRUPT RECEIVER ROUTINE



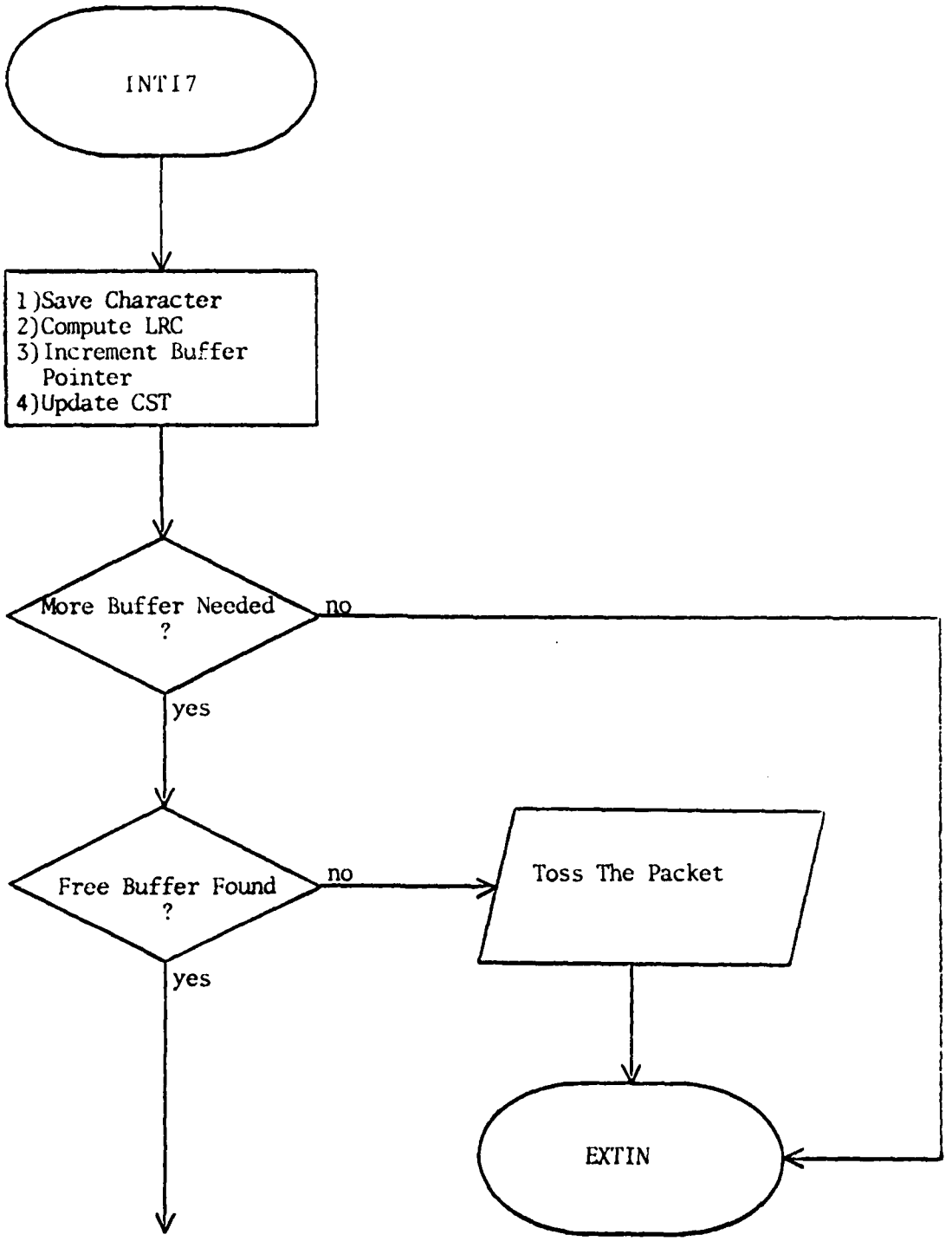
Interrupt Receiver Routine, Con't



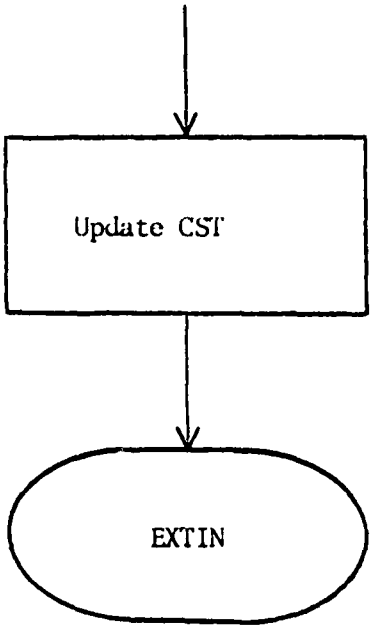
Interrupt Receiver Routine, Con't



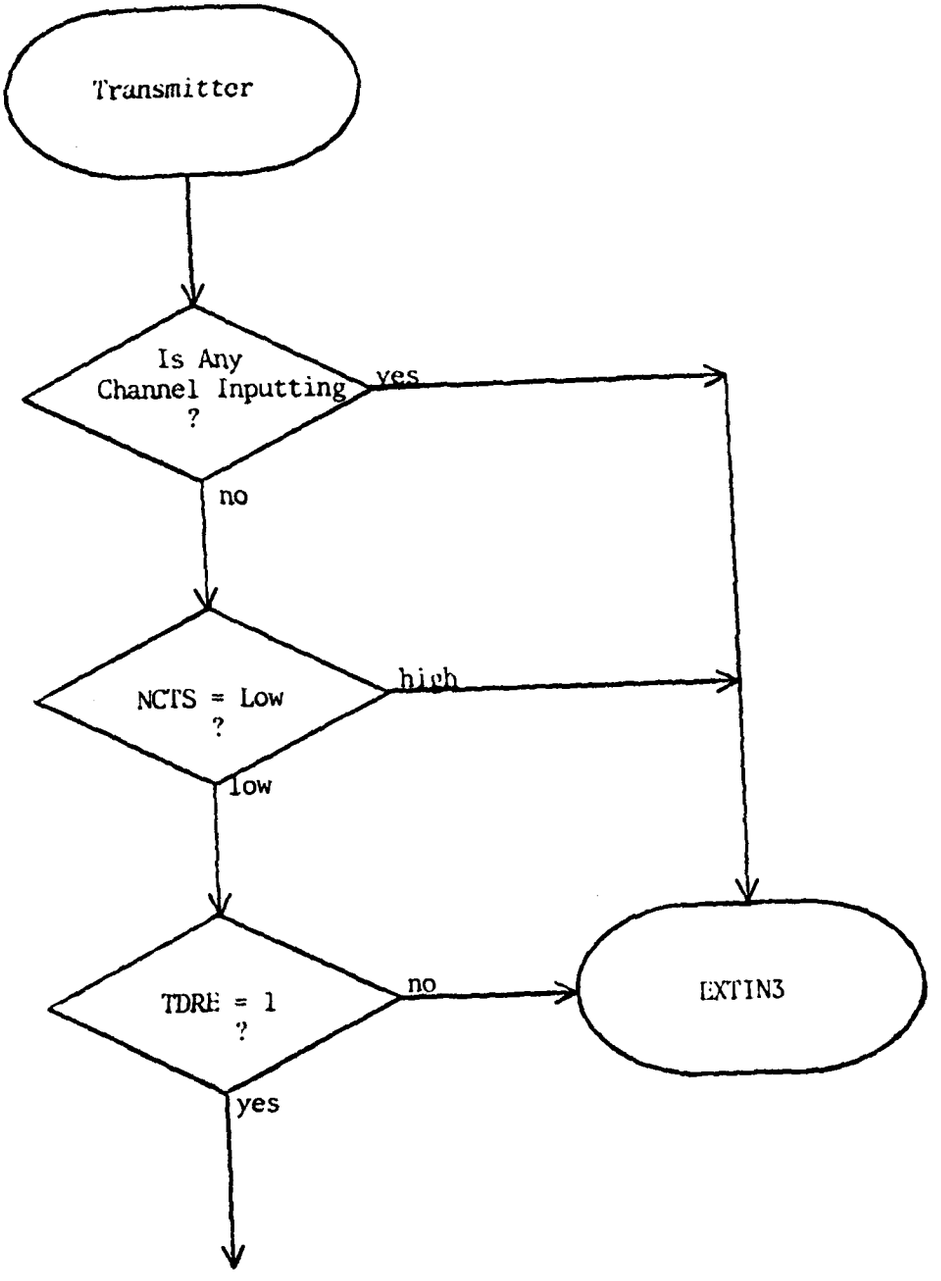
Interrupt Receiver Routine, con't



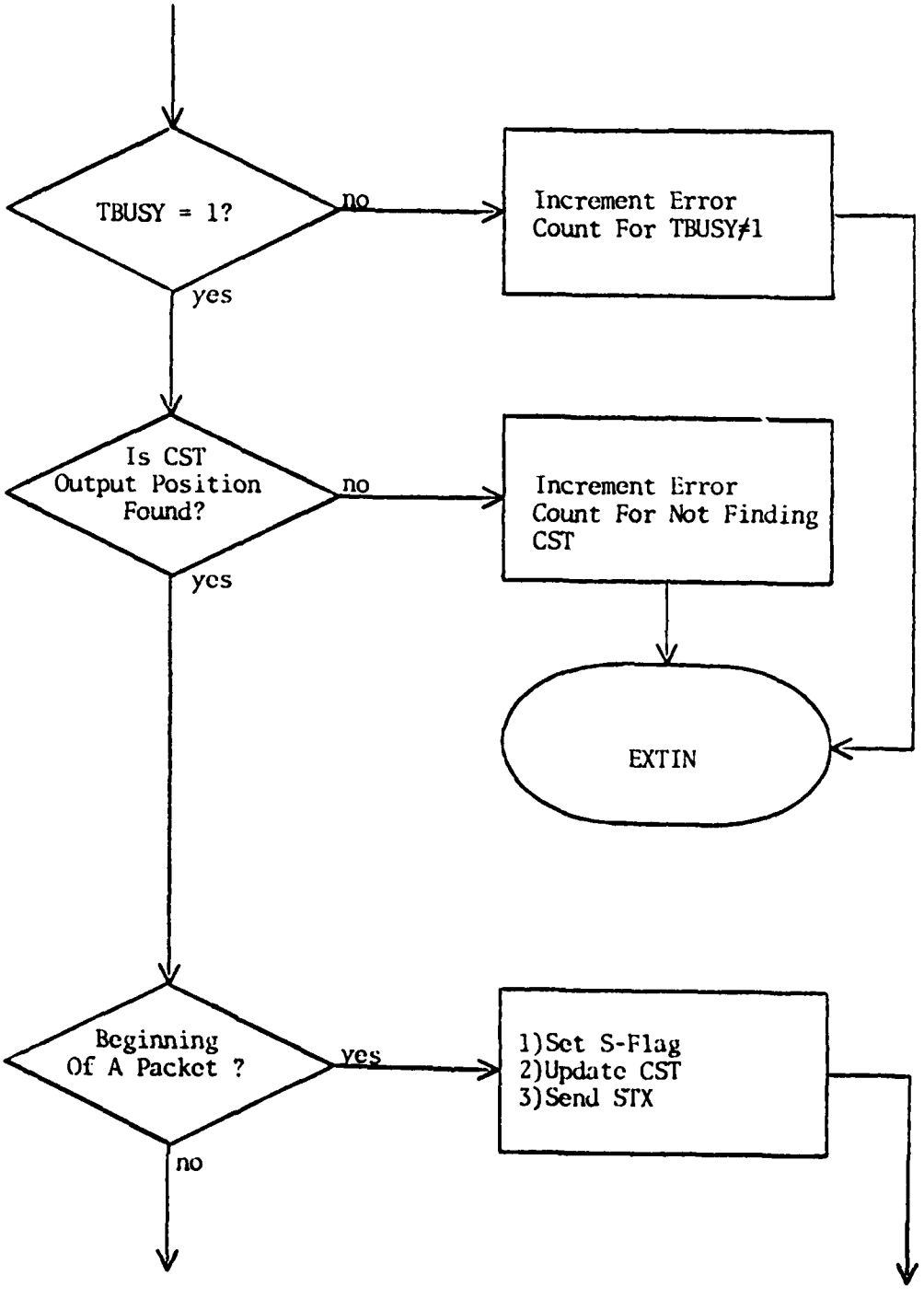
Interrupt Receiver Routine, con't



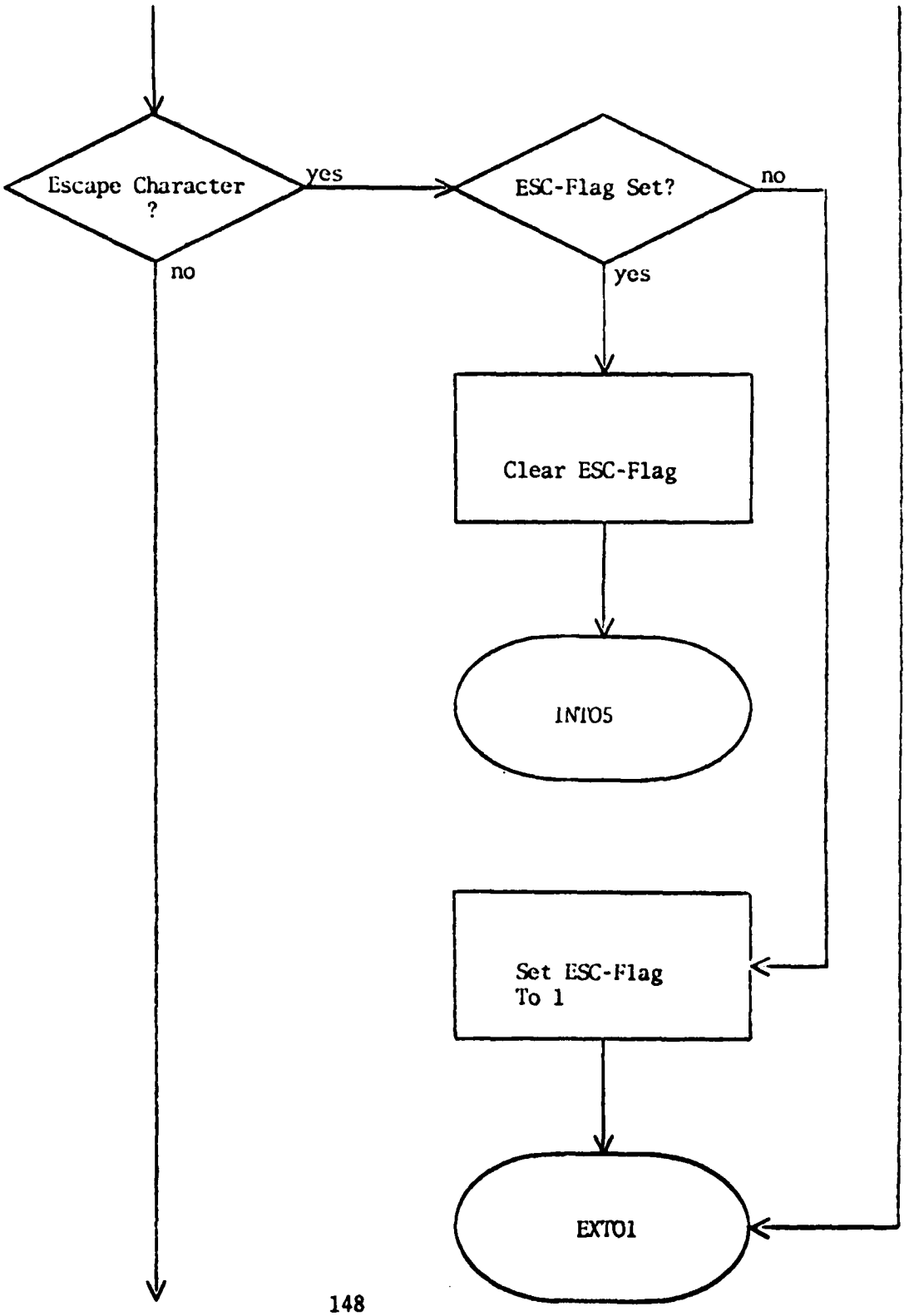
INTERRUPT TRANSMITTER ROUTINE



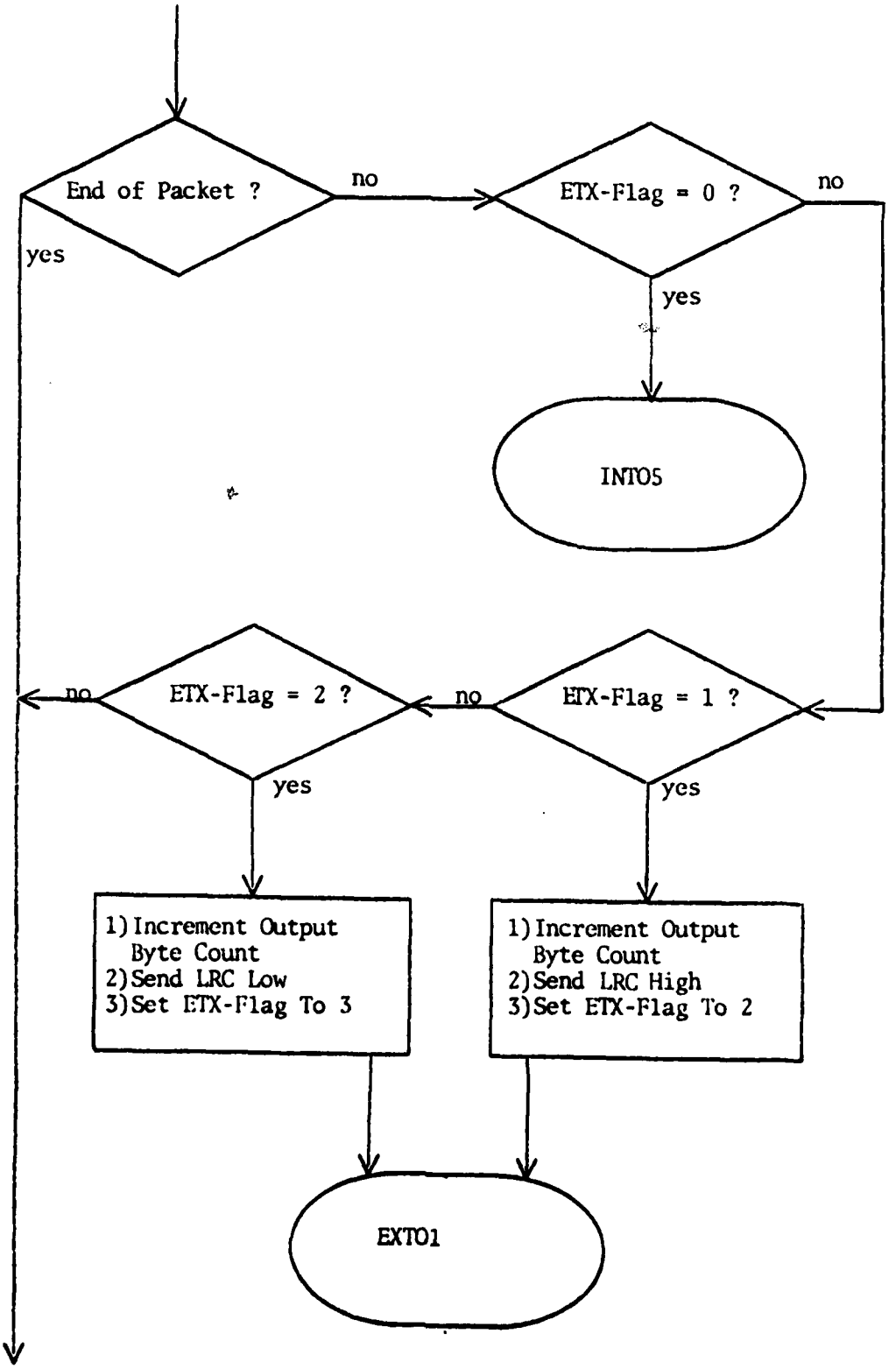
Interrupt Transmitter Routine, con't



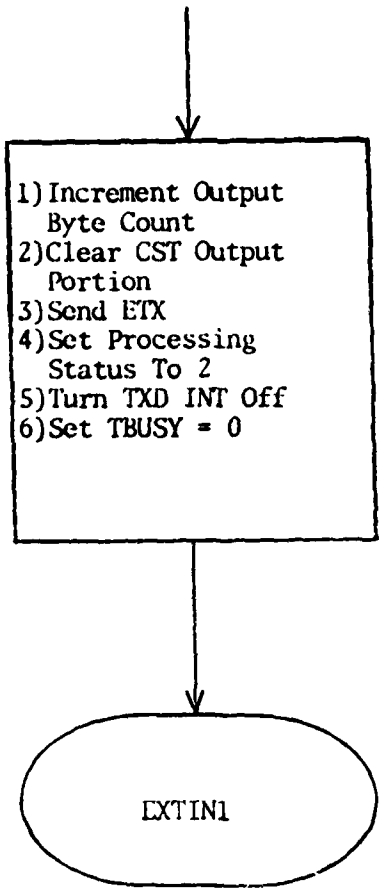
Interrupt Transmitter Routine, con't



Interrupt Transmitter Routine, con't

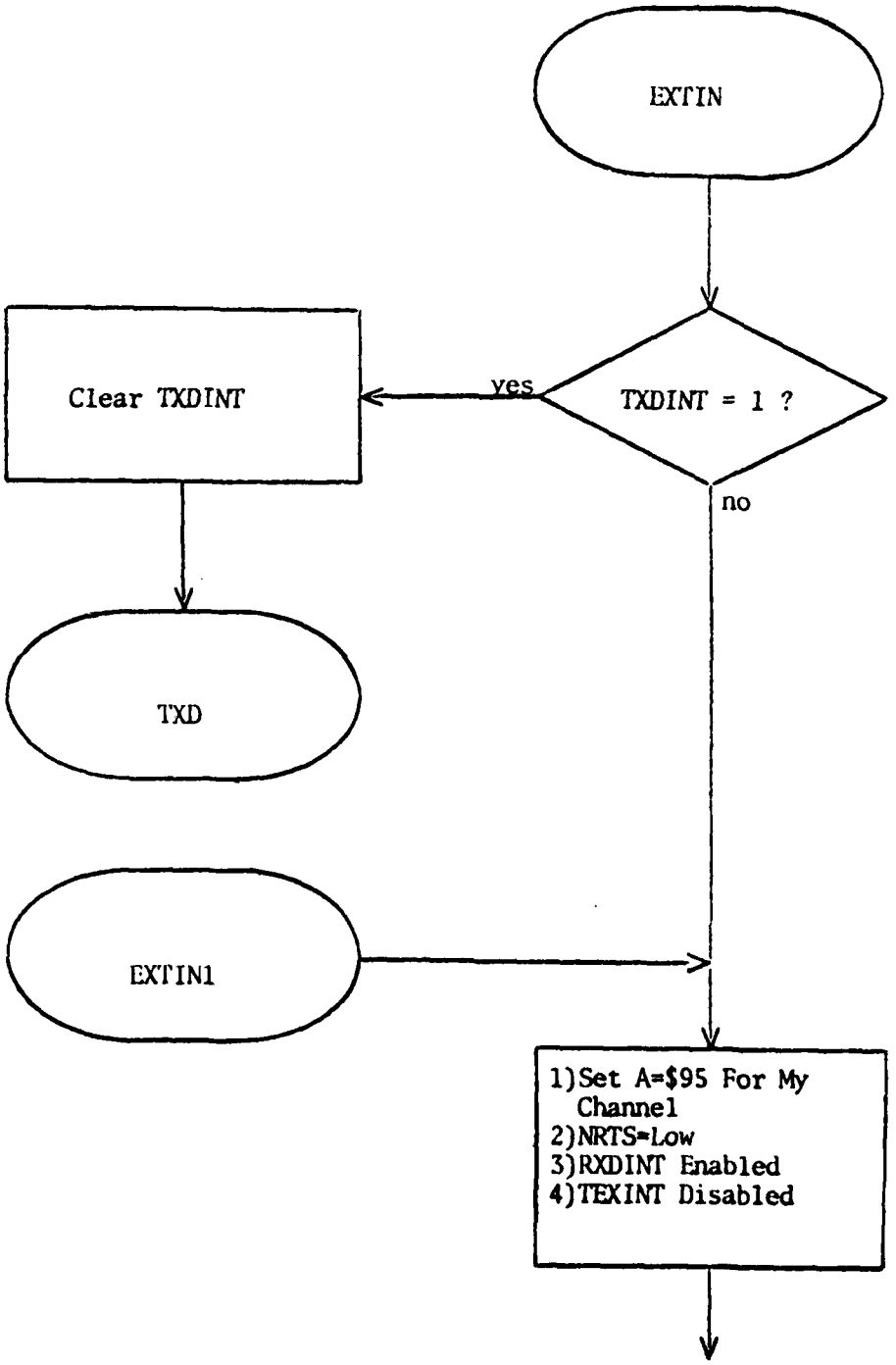


Interrupt Transmitter Routine, con't

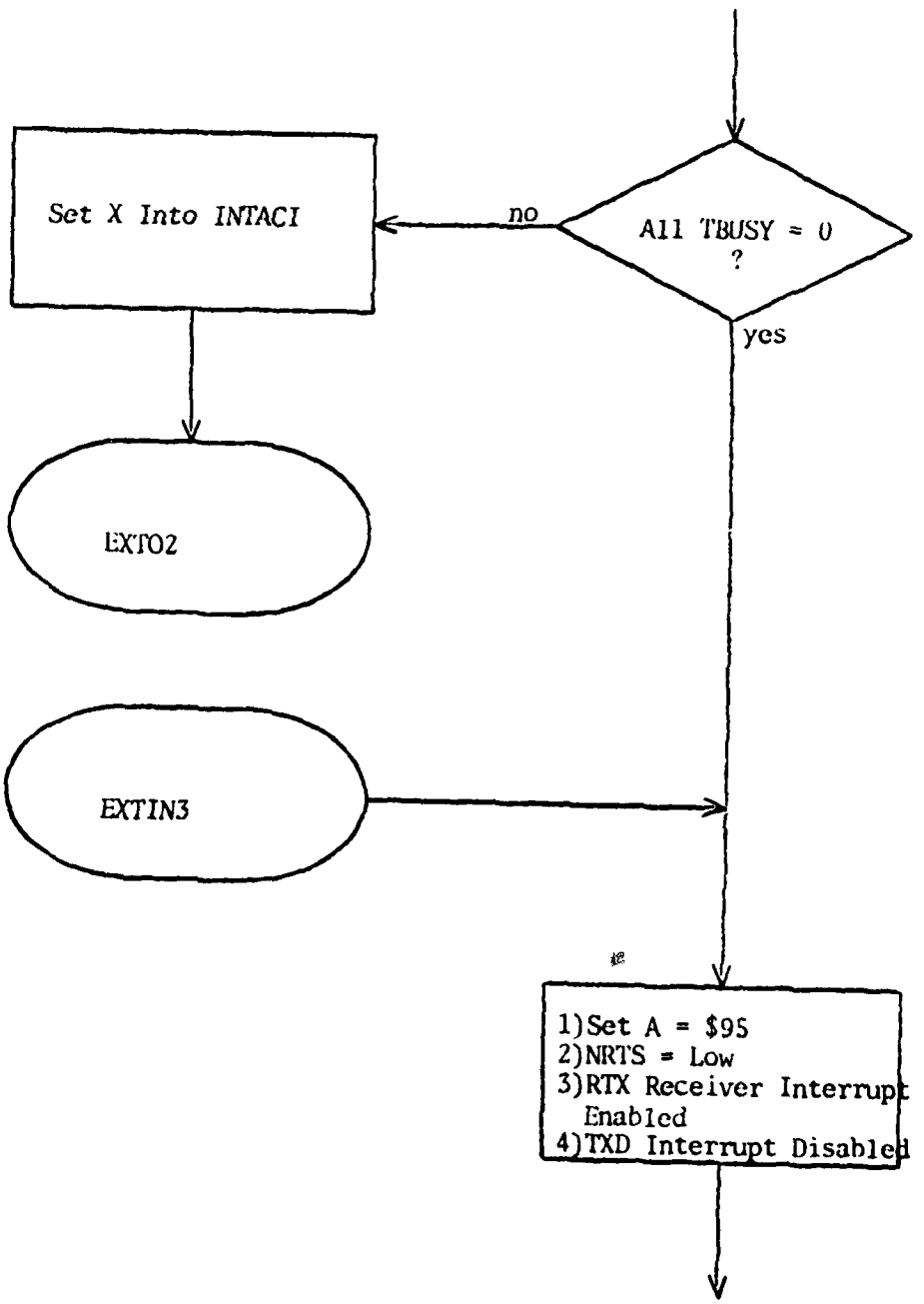


RE-ENTRANT INTERRUPT EXIT ROUTINE (MULTIPLE ENTRY POINTS)

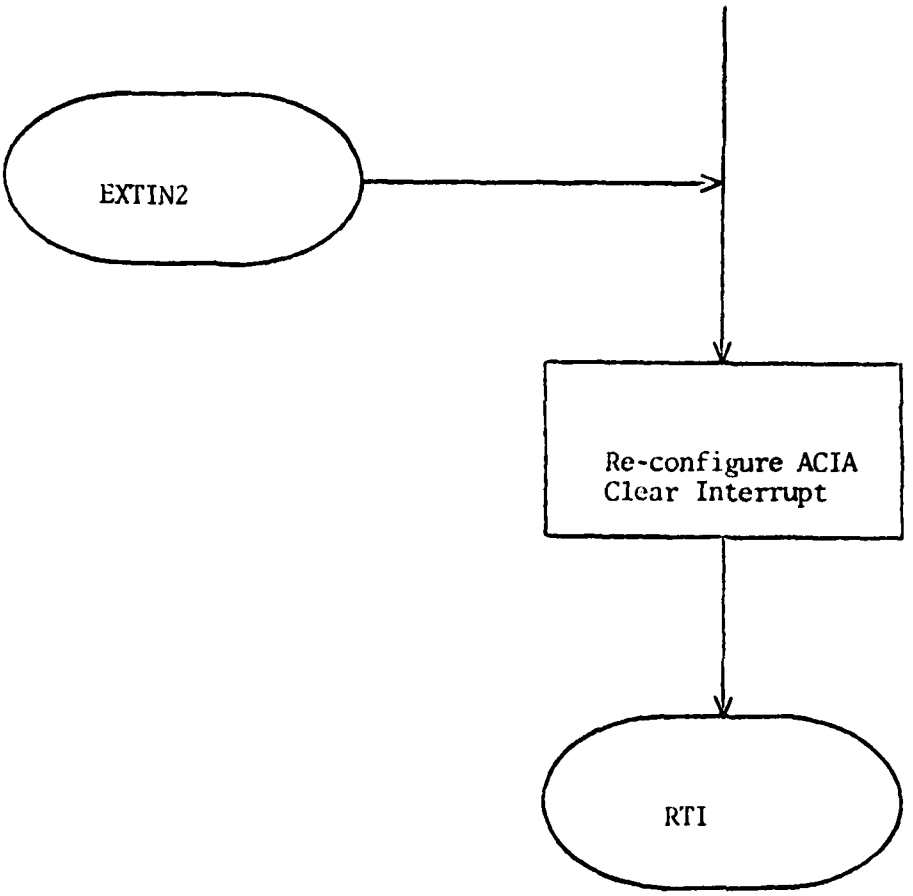
Input Interrupt Exit



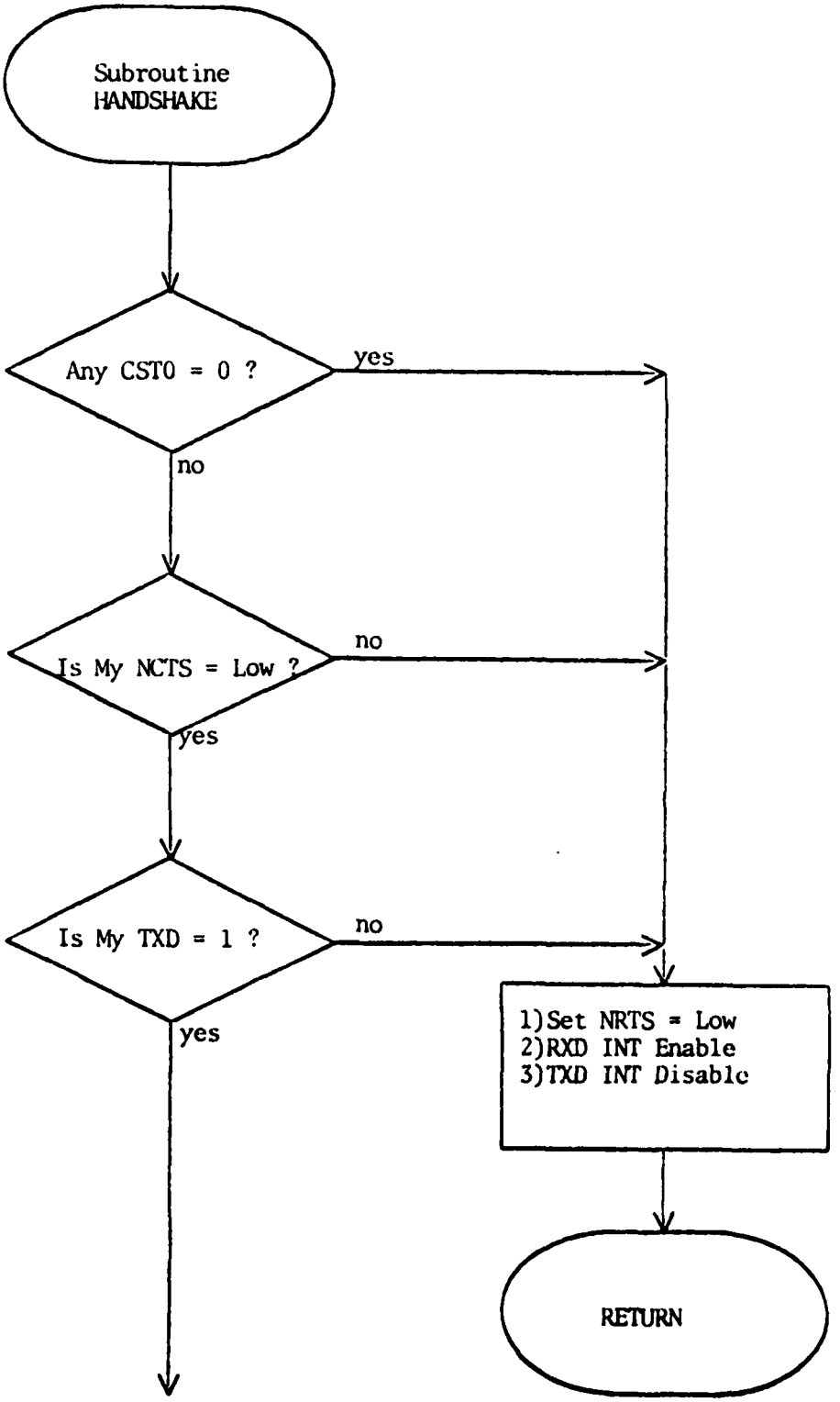
Input Interrupt Exit, con't



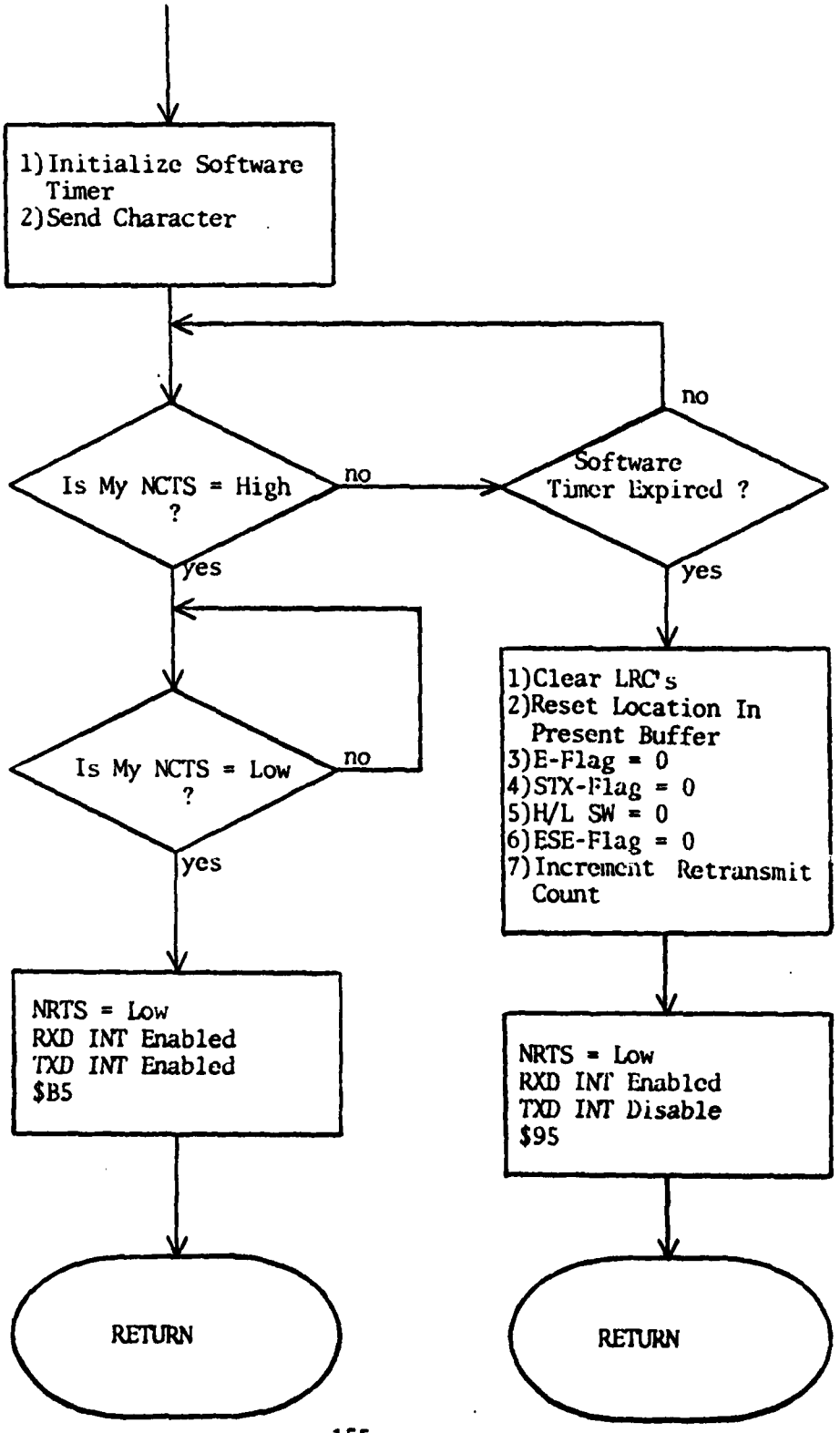
Input Interrupt Exit, con't



SUBROUTINE HANDSHAKE



Subroutine HANDSHAKE, con't




DATE
ILME